A BIOINFORMATIC TOOL FOR ANALYSING THE STRUCTURES OF PROTEIN COMPLEXES BY MEANS OF MASS SPECTROMETRY OF CROSS-LINKED PROTEINS

by

Shannon L.N. Mayne

Supervisor: Prof. Hugh -G. Patterton

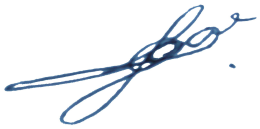
A Dissertation in fulfilment of a Masters of Science degree in Biochemistry

University of the Free State

2013

DECLARATION

I declare that the dissertation hereby submitted for the *Magister Scientiae* degree at the University of the Free State through the Faculty of Natural and Agricultural Sciences is my own work and has not been previously submitted by me at another University for any degree. I cede copyright of this dissertation in favour of the University of the Free State.



_____

Shannon Leon Noël Mayne

January 2013

ACKNOWLEDGEMENTS

Thanks to staff and students at the University of the Free State, in particular: Pankaj Sharma and Gabre Kemp for experimental assistance, as well as Leon du Preez and the UFS ICT Services staff for assistance with the server access and settings. Special thanks are extended to Professor Hugh Patterton for invaluable input, guidance and indefatigable patience. Heartfelt gratitude and appreciation go to my family and closest friends for their unstinting support and understanding throughout. A postgraduate bursary from the former National Bioinformatics Network (NBN) is also gratefully acknowledged.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

Chapter 1

Literature review: Bioinformatics tools for the structural elucidation of multi-subunit protein complexes by mass spectrometric analysis of protein-protein cross-links.

1.1 Introduction

Supra-molecular protein complexes are involved in numerous fundamental biochemical processes including catalysis, protein secretion, nuclear transport, protein degradation, protein folding, gene regulation, RNA synthesis, protein synthesis, signal transduction, chromosome segregation, and in DNA replication and repair. A mechanistic understanding of these composite protein assemblies requires an insight into the molecular arrangement and interactions of the sub-units. A structural insight into the arrangement of the components in such complexes is, however, still limited. Traditional methods of protein structure determination such as X-ray crystallography (XRC) and Nuclear Magnetic Resonance (NMR) have technical limitations, restricting the size of the protein complex that can be crystallized, or the resolution at which large structures can be interpreted. Cryo-electron microscopy holds great promise for the structural elucidation of mega-Dalton protein complexes, but the resolution is currently insufficient for a detailed structural analysis (Jonic and Venien-Bryan, 2009).

Recently the application of mass spectrometry (MS) to identify the positions of chemical cross-links between the protein sub-units of complexes has significantly advanced our understanding of the arrangement and interaction surfaces involved in mega-Dalton protein complexes (McHugh and Arthur, 2008a). If the structures of the components are known, the location of the cross-links allows one to very precisely place each sub-unit in the correct orientation within the complex. The use of a range of cross-linking reagents, each with a specific atomic reach, has allowed the further refinement of models of quaternary protein structures. This approach, termed “MS3D”, has developed into a powerful technique for the structure elucidation of multi-subunit protein complexes.

Many software packages are available for standard protein identification or *de novo* sequencing analyses by MS (McHugh and Arthur, 2008b). Software for the analysis of MS and MS/MS spectra to identify cross-linked peptides and the sequence positions of the residues involved in the cross-link has been developed for very specific

experimental methodologies. In this review we provide an overview of software currently available for MS3D analyses, noting the application niches of programs as well as giving a detailed comparison of their functionalities. This gives an overview of the current software landscape as well as facilitating selection of the most suitable tool for a particular MS3D application. For the basic mass spectrometric concepts of this technique we refer the reader to a number of recent reviews (Breddam and Meldal, 1992b; Kang *et al.*, 2009a; Leiros *et al.*, 2004b).

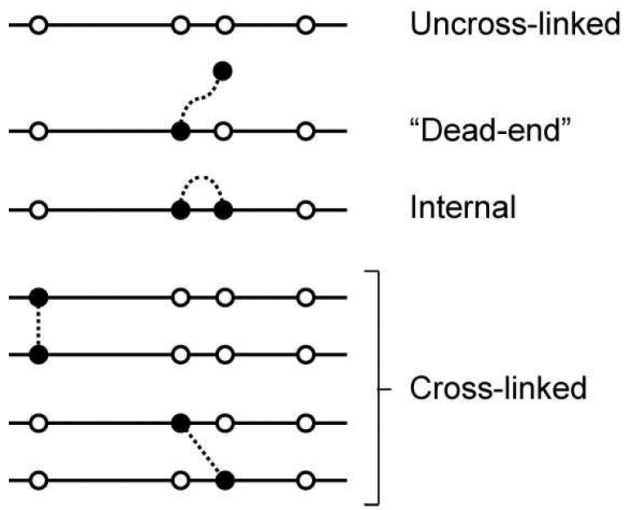


Figure 1.1: The different types of cross-linked peptides that may be generated following proteolytic cleavage of chemically cross-linked proteins in a complex are shown. Reactive amino acid side-chains are indicated by the white circles. The black circles connected by a dotted line indicate a bi-functional cross-linking reagent. Where multiple reactive amino acid side-chains were within cross-linking reach in the original protein complex, combinations of cross-linked di-peptide isomers can form.

1.2 MS3D data analysis

In a typical MS3D experiment a protein complex is isolated or reconstituted and then treated with a cross-linking reagent. A wide variety of cross-linking reagents are commercially available, with homo- and bi-functional reactive groups targeting a specific or a narrow range of residue side-chains (Wong, 2010).

Some of these reagents allow subsequent cleavage or affinity purification (Kang *et al.*, 2009b). Following cross-linking, the complex is digested with a sequence specific protease such as trypsin (Leiros *et al.*, 2004a) or endo gluC (Breddam and Meldal, 1992a). This yields a mixture of (possibly multiply) cross-linked, singly linked ("dead-end"), intra-linked and uncross-linked peptides (Figure 1.1).

Cross-linked peptides are typically identified by MS, often followed by confirmation with MS/MS. The residues involved in the cross-link in the identified peptides are usually pinpointed by MS/MS. Software used for the MS-based structural elucidation of cross-linked protein complexes normally perform four steps: 1) detection of the cross-linked peptides, 2) identification of the cross-linked peptides, 3) identification of cross-linked residues in the di-peptide, and, 4) interpretation of cross-linked data in terms of spatial proximities and subsequent refinement of the structural model (Figure 1.2). We discuss each of these four steps individually, mentioning the different experimental routes that have been reported, and indicate the applicability of programs to each of the various approaches. A summary of the abilities of each program is presented in Table 1.1. No single software program is currently available that can perform all four tasks. In particular, the interpretation of spatial constraints and model refinement still require significant manual input. The reader should also take note of the very similar naming of X!Link, X-Link, X-Links, XLINK, Links/MS2Links and SearchXLinks. These are all distinct programs.

1.2.1 Detection of cross-linked peptides

The first activity in a MS3D experiment is the chemical cross-linking of the proteins in the biological complex (Ong *et al.*, 2002b; Schnolzer *et al.*, 1996a; Takao *et al.*, 1991a; Tang *et al.*, 2005e). The yield of chemically cross-linked peptides under conditions that conserve the structural integrity of a biological complex is often very low (Kang *et al.*, 2009c) and the detection of the di-peptides in a complex mass spectrum can therefore be technically challenging. Four approaches have been reported to simplify the identification of the cross-linked peptide peaks in the mass spectrum (Step 1, Fig 2): comparison to a non-cross-linked control, using isotopically labelled di-peptides, identification of post-fragmentation reporter ions, and the identification of peaks that match the theoretical mass of one of the possible peptide dimer combinations that can be formed from the known proteins in the complex.

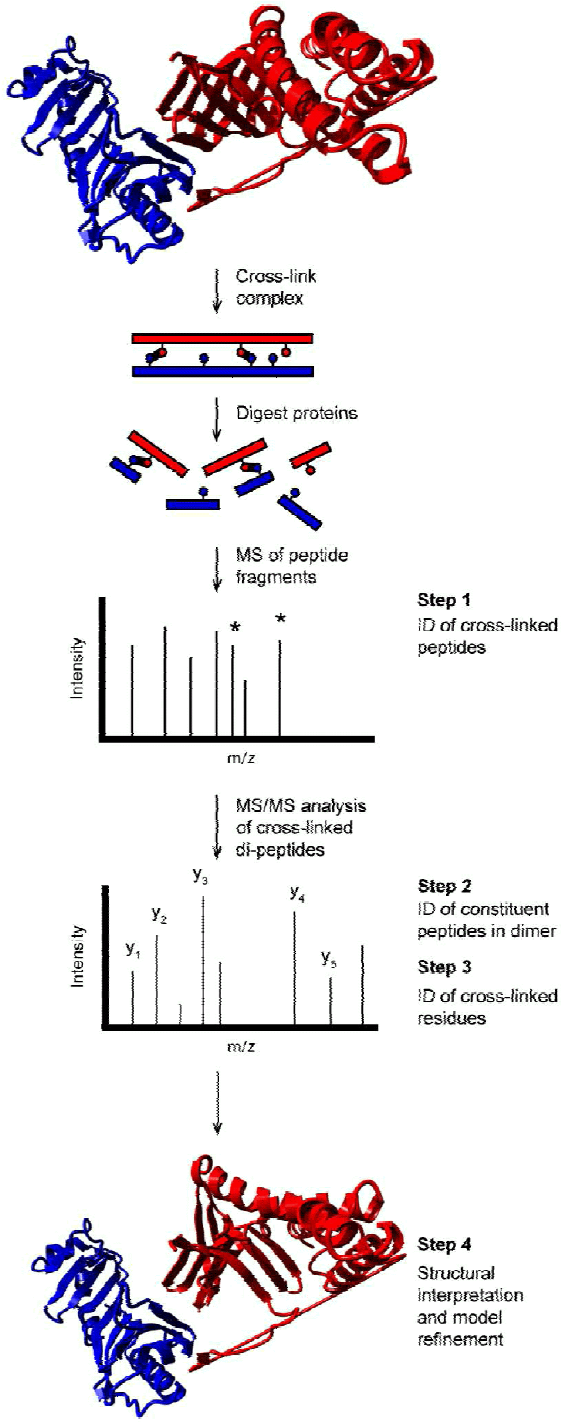


Figure 1.2: The workflow involved in determining the orientation and relative positioning of the sub-units in a composite protein complex by MS3D is shown. Cross-linked di-peptides are typically identified by MS (peaks denoted by asterisks), and should be verified by MS/MS. The constituent peptides in a cross-linked di-peptide as well as the reactive amino acid side-chains involved in the cross-link are identified by MS/MS. Several different experimental approaches to simplify identification of di-peptides are mentioned in the text. The main steps that are supported by the various bioinformatics software packages are indicated.

Table 1.1: The characteristics, experimental requirements and computational approaches of different MS3D bioinformatics tools.

Software	Ref. [g]	Sample preparation[a]	Detect x-linked peptide[b]	Seq of x-linked peptide[c]	ID of x-linked res.	Distance limits[d]	Struct. inter-pretation[e]	Score[f]
ASAP & MS2Assign	42, 60	None	None	MS & MS/MS	Yes	No	None	NP
CLPM	52	Non-x-linked control	Type ID MS	MS	No	No	None	None
CrossSearch	30	Single-protein x-linked control	Type ID MS	MS & MS/MS	Yes	No	None	None
Crux	29	None	None	MS & MS/MS	Yes	No	None	P
Links & MS2Link	20	None	None	Top down MS/MS	Yes	No	None	None
MassMatrix	57	None	None	MS & MS/MS	Yes	No	None	P
MS2PRO	22	None	None	Top down MS/MS	Yes	No	None	NP
MS-Bridge	9	None	None	MS	No	No	None	None
MSX-3D	14	None	None	MS	No	Yes	Validation	None
PeptideMap (in PROWL)	12	Non-x-linked control	Type ID MS	MS	No	No	None	None
Pro-Crosslink	13	$H_2{}^{18}O$ isotope labeling	Type ID MS	MS & MS/MS	Yes	No	None	NP
ProteinXXX / GPMAW	31, 36	None	None	None	No	No	None	None
SearchXLinks	43	Non-x-linked control	MS PSD	MS & MS/MS & PSD	Yes	No	None	NP
VIRTUAL-MSLAB	11	None	None	MS	No	No	None	None
X!Link	23	None	None	MS & MS/MS	Yes	No	None	NP
xComb	34	None	None	None (additional software)	No (additional software)	No	None	None
X-Link	53	MIX isotope labelling	Type ID MS	MS	No	No	None	None
XLINK (iXLINK & doXLINK)	46	$H_2{}^{18}O$ & ^2H isotope labelling; only NHS	Type ID MS	MS & MS/MS	Yes	No	None	P
X-Links	2	None	Type ID MS/MS	MS & PIR	No	No	None	NP
xQUEST	38	^2H isotope labelling	Type ID MS/MS	MS & MS/MS	Yes	No	None	NP & P

[a] Entries indicate the requirement of software for any specific cross-linking reagent, labelling method, or control sample. MIX refers to analysis of mixed isotope samples (Ong *et al.*, 2002a), and NHS to N-hydroxysuccinimide based cross-linking reagents.
[b] Entries indicate whether the software can identify a cross-linked peptide in the MS spectrum. "Type ID" identifies software that is capable of discriminating between different cross-link types. PSD: post-source decay.
[c] The type of experimental data that the software requires to identify the sequences of the cross-linked peptides. PSD: post-source decay, PIR: protein interaction reporter.
[d] Indicates whether the software provides any limit on the maximum distances between cross-linked residues.
[e] Indicated whether the software performs any interpretation of the cross-link data in terms of the structure of compound protein assemblies.
[f] P: probabilistic, a statistical probability; NP: non-probabilistic, a score relative to a threshold value.
[g] References 61-63 are first cited in this table (Nielsen *et al.*, 2007; Panchaud *et al.*, 2010; Peri *et al.*, 2001).

1.2.1.1 Non-cross-linked controls

In the simplest approach, peaks that are present in the MS spectrum of a cross-linked sample and absent in an uncross-linked control sample are flagged as putative cross-linked di-peptides by CLPM (Tang *et al.*, 2005d), PROWL's PeptideMap (Fenyo, 1997) and SearchXLinks (Wefing *et al.*, 2006a). In a variation of this technique, CrossSearch (Nadeau *et al.*, 2008) identifies only inter-molecular cross-linked di-peptides by analysing peptides from the cross-linked complex as well as the two sub-units cross-linked individually, which requires analysis of at least three cross-linked samples.

1.2.1.2 Isotope labelling

The isotopic labelling of cross-linked peptides is achieved by using a cross-linking reagent that was synthesized using compounds that contained heavy or light isotopes (Takao *et al.*, 1991b), the introduction of ^{18}O from $H_2{}^{18}O$ during proteolytic hydrolysis of the proteins (Schnolzer *et al.*, 1996b), or by the cross-linking of mixed isotope samples (MIX), typically prepared from cells that were cultured in media that contained ^{14}N or ^{15}N labelled amino acids (Ong *et al.*, 2002c).

Cross-linking with a mixture of heavy and light cross-linking reagent will produce peak pairs or doublets in the MS spectrum that are offset by the mass difference between the two isotopically labelled reagents, and with intensities that reflect the ratio of heavy:light reagent. Thus, cross-linked peptides can easily be found by scanning the mass spectrum for these peak pairs with a specific mass difference. GPMAW (Anderson *et al.*, 2007a; Schnaible *et al.*, 2002a), iXLINK (Seebacher *et al.*, 2006b) and xQUEST (Rinner *et al.*, 2008d) can scan MS spectra to identify peak doublets and potential cross-linked di-peptides. In GPMAW and iXLINK a custom mass difference in the peak pair of up to 8 Da can be selected. iXLINK can also identify single peptides that contain "dead-end" links by detecting peak doublets that appear for each of the two isotopically labeled cross-linking reagents following hydrolysis of the single unreacted functional group in a $H_2{}^{16}O/H_2{}^{18}O$ mixture.

Trypsin incorporates two oxygen atoms from two water molecules at each carboxyl terminal during hydrolysis (Ye *et al.*, 2009). If proteolytic cleavage is performed separately in $H_2{}^{16}O$ and in $H_2{}^{18}O$, and the samples combined before MS analysis, an 8 Da mass differences will be visible in peak pairs of peptide dimers (Back *et al.*, 2002). Doublets separated by 8 Da are highlighted by DetectShift in Pro-Crosslink (Gao *et al.*,

2006b). Peptide dimers that contain one or both of the original C-termini of the protein will not be labeled by ^{18}O, and will thus not be flagged in this method.

In the mixed isotope procedure (MIX), proteins purified from cells grown in the presence of ^{14}N and ^{15}N labeled amino acids are combined in equal parts. Intra-molecular cross-linked peptides will be detected as peak doublets composed of N^{15}/N^{15} and N^{14}/N^{14} dimers. Inter-molecular dimers, on the other hand, will be observed as peak triplets composed of N^{15}/N^{15}, N^{14}/N^{15}, and N^{14}/N^{14} linker peptides. X-Link (Taverner *et al.*, 2002a) uses these isotope signatures to detects inter-molecularly cross-linked peptides.

1.2.1.3 Post-fragmentation reporter ions

This approach requires the use of either a disulphide or a special type of cross-linker, termed a "protein interaction reporter" (PIR) (Tang *et al.*, 2005a). In the case of disulphide bonds, in-source decay often results in the loss of (H_2 + H^+) (Schnaible *et al.*, 2002b). SearchXLinks identifies peptides linked by a disulphide bridge by searching for a group of three peaks, where the combined mass of two peaks equals that of the third, minus the mass of (H_2 + H^+) (Wefing *et al.*, 2006b). X-Links (Anderson *et al.*, 2007b) is specifically designed to use with a PIR cross-linking reagent (Tang *et al.*, 2005b). PIR linkers fragment during collision induced dissociation (CID) in a defined fashion, releasing a signature reporter ion and the modified single peptides. Where X-Links detects the PIR-derived reporter ion in an MS/MS spectrum, the spectrum is further scrutinized to identify two peaks where the combined mass plus that of the reporter ion matches a peak in the MS spectrum, which is then flagged as a cross-linked peptide. This approach was successfully used to identify interaction partners and interaction sites *in vivo* (Zhang *et al.*, 2009).

1.2.2 Matching peaks to a library of possible peptide dimers

Where the proteins in a complex are known, many programs follow the route of creating a library of all possible cross-linked peptides based on the specificity of the proteolytic enzyme selected, the chemical composition of the cross-linking reagent, the reactive amino acid residue, and the allowed post-translational modifications selected by the user. Peaks in the experimental MS spectra that match the mass of entries in this library are flagged as possible cross-linked di-peptides. The programs ASAP (Singh *et al.*, 2008b), CLPM, MS-Bridge (Clauser *et al.*, 1999), MSX-3D (Heymann *et al.*, 2008a), PROWL's PeptideMap, VIRTUALMSLAB (de Koning *et al.*, 2006a) and X-

Link generate a list of such matched peaks. Some programs can make use of data from subsequent MS/MS analyses of flagged peptides to confirm the presence of the di-peptide (Crux (McIlwain *et al*., 2010), MS2Assign (Schilling *et al*., 2003), SearchXLinks, Pro-Crosslink, iXLINK, X-Links, GPMAW, X!Link (Lee *et al*., 2007), CrossSearch and xQUEST). Programs such as MS2Links (Kellersberger *et al*., 2004) and MS2PRO (Kruppa *et al*., 2003) that are used in a top-down proteomics approach omit the initial MS step, using only data from the MS/MS analysis.

Several groups (Chu *et al*., 2010; Maiolica *et al*., 2007; Singh *et al*., 2008) have used existing peptide search engines intended for single peptides, such as MASCOT (Perkins *et al*., 1999) or X!Tandem (Craig and Beavis, 2004), to match the experimental spectra of cross-linked peptides.

1.2.3 Generating the library of peptide dimers

The degree to which the user can customize the theoretical library of possible cross-linked peptides, in terms of the allowed post-translational modifications or the chemical composition and residue specificity of the cross-linking reagent for example, differ between software packages (summarized in Table 1.2). This limits the type of experimental data that each software tool can analyse. SearchXLinks does not support any user-defined modification types. ASAP, CLPM, MS2Assign, MS2Links, MSX-3D, VIRTUALMSLAB, SearchXLinks, iXLINK and xQUEST allow a limited number of post-translational modifications. CLPM permits a maximum of ten custom modification types. MS2Assign, MS2Links, MSX-3D, VIRTUALMSLAB, iXLINK and xQUEST, in contrast, allow the inclusion of any number of modifications. ASAP, CLPM, CrossSearch, GPMAW, MassMatrix (Xu *et al*., 2008), MS2Assign, MS2Links, MS2PRO, MSX-3D, PROWL's PeptideMap and Pro-Crosslink allow the user to specify custom cross-linking reagents and reactive amino acid side-chains. As a novel ability, XLINK also provides for custom amino acids, but was developed solely for the amine-specific, CID-cleavable PIR cross-linker. MassMatrix, ProteinProspector's MS-Bridge and PROWL's PeptideMap are applicable only to disulphide bridge cross-links.

Table 1.2: Parameters included in the theoretical MS and fragmentation libraries of different software packages.

Software	PTMs[a]	Cross-linker[b]	Cross-link type[c]	Protease[d]	Sequences[e]	Fragmentation[f]	Other[g]
ASAP & MS2Assign	Any no. Custom	Custom (1)	0, 1, 2	Select	1	a, b, c, x, y, z; NH_3, H_2O, CO, CO_2 loss; immonium	
CLPM	Any no. Any type Custom	Custom (□10)	1, 2	Custom	2	None	
CrossSearch	None	Select (4)	0, 2	NS	2	None	for FTICR-MS
Crux	None	NS	0, 1, 2	NS	Any no.	b, y; H_2O, NH_3, CO loss	
Links & MS2Link	Any no. Any type Custom	Select	2	NA	1	SORI-CID; MS^n; Nucleic acids(a-B,d-H_2O,w,y); internal fragments; a, b, c, x, y, z; CO, NH_3, H_2O, CO_2 loss; immonium	Isotope pattern filter; nucleic acids
MassMatrix	Any no. Any type	SS bridges	NA	Select	Any no.	NS; H_2O, NH_3 loss; Custom rules	
MS2PRO	None	Custom	0, 2	NA	1	b, y; internal ions	
MS-Bridge	Any type	SS bridges	NA	Select	1	None	
MSX-3D	Any no. Any type Custom	Select / Custom	2	Select / Custom	Any 3	None	Any no. peptide chains
PeptideMap (in PROWL)	Any type Custom	SS bridges	NA	Custom	1	None	
Pro-Crosslink	None	Custom	2	Trypsin	2	a, b, c, x, y, z; H_2O, NH_3, CO_2, CO loss; double fragmentation	Custom amino acid
ProteinXXX / GPMAW	None	Custom	0, 1, 2	Select	2	a, b, c, x, y, z; H_2O, NH_3 loss	
SearchXLinks	Any type	Select	2	NS	NS	ISD; CID: a, b, c, x, y, z; proline effect	
VIRTUAL-MSLAB	Any type Custom	Select	1, 2	Select	Any no.	None	
X!Link	None	NS	0, 2	Trypsin	2	b, y; double fragmentation	
xComb	None	NA	NA	Select	Any no. (<50)	None	Exclude peptides with too few miss cleavages
X-Link	None	NS	0, 2	Trypsin	2	None	
XLINK (iXLINK & doXLINK)	Any no. Any type Custom	NS	0, 1, 2	Custom	NS	b, y; H_2O, NH_3, CO_2, CO loss; Reporter	Custom amino acid
X-Links	None	PIR	0, 1, 2	Custom	Any no.	None (only MS/MS reporter ion)	
xQUEST	Any no. Any type Custom	Select/Custom	2	Select / Custom	Any no.	Ion-tag mode: b,y; Enumeration mode: NS	

[a] The entries show the number of post-translational modifications (PTMs) and the number of different PTM types that are allowed per di-peptide, as well as whether custom PTMs can be defined.

[b] The different cross-linking reagents that can be screened for. NS: not stated; NA: not applicable. Numbers in brackets indicate the maximum number that can be selected or defined. PIR: protein interaction reporter.
[c] Different types of cross-linked peptides that can be screened for. 0: "dead end" link, 1: singly linked; 2: cross-linked di-peptide.
[d] Compatibility of different protease with analysis software. NS: not stated; NA: not applicable (only analyses non-digested samples).
[e] Number of sequences, and, by implication, number of protein sub-units in the composite protein complex that various software packages can analyse.
[f] Fragmentation ion-types and losses that software packages can analyse. NS: not stated. PIR: protein interaction reporter, ISD: in-source decay.
[g] Entries indicate additional capabilities of the various software packages.

Crux, Pro-Crosslink, X-Link and X!Link assume the use of trypsin as a protease, and does not allow selection of a different cleavage enzyme. CLPM, GPMAW, PROWL's PeptideMap, X-Links and iXLINK, on the other hand, support any protease with a defined target sequence. As a further refinement, PROWL's PeptideMap will allow proteases that cleave the bond on the N-terminal (eg. Thermolysin (Ambler and Meadway, 1968)) as well as C-terminal side of the residue recognized by the protease.

An important consideration when comparing experimental MS spectra of cross-linked peptides to a library of all possible pairs is the presence of various combinations of cross-linked species in the mass spectrum (see Fig.1.1). Crux, MS2Assign, X-Links and iXLINK can accommodate different classes of cross-linked peptides. CrossSearch, GPMAW, X-Link and X!Link will flag dead-end linkers, while CLPM and VIRTUALMSLAB also allow identification of intra-molecular cross-links.

In many studies reported to date cross-linking was carried out on complexes formed by only two proteins with known sequences (Balasu *et al*., 2009; Chu *et al*., 2004; Pagnozzi *et al*., 2010; Pimenova *et al*., 2008). Crux, MassMatrix, MSX-3D, VIRTUALMSLAB, X-Links and xQUEST will accept any number of protein sub-units in the studied complex. Other programs accept only one (ASAP, PROWL's PeptideMap, MS2Links and MS2PRO) or two (CLPM, CrossSearch, GPMAW, Pro-Crosslink, X-Link, X!Link) proteins. Nucleic acids also form part of many protein complexes such as in chromatin, ribosomes and snRNPs. The theoretical libraries generated by CLPM and MS2Links can include both protein and nucleic acid sequences.

1.2.4 Matching experimental peaks to the theoretical library

All the programs reviewed here will score two peaks as a match when the *m/z* values of an observed peptide peak and a theoretical peak are within a specified range. Some programs match the experimental peaks to the theoretical library, ignoring absent experimental peaks (ASAP, CLPM, Crux, Pro-Crosslink, VIRTUALMSLAB, X-Links,

iXLINK and xQUEST) (Anderson *et al.*, 2007c; de Koning *et al.*, 2006b; Gao *et al.*, 2006a; McIlwain *et al.*, 2010d; Rinner *et al.*, 2008c; Seebacher *et al.*, 2006a; Tang *et al.*, 2005c; Young *et al.*, 2000) whilst others match each theoretical peak to the list of experimental peaks (MS2Assign, MSX-3D, SearchXLinks, X-Link, X!Link) (Heymann *et al.*, 2008b; Lee *et al.*, 2007a; Schilling *et al.*, 2003a; Taverner *et al.*, 2002b; Wefing *et al.*, 2006c). Software can search for either both average and mono-isotopic masses (ASAP, GPMAW, MS2Assign and MSX-3D) or only for mono-isotopic masses (CLPM, CrossSearch, MS2Links, Pro-Crosslink, SearchXLinks, VIRTUALMSLAB, iXLINK and xQUEST). In the experience of the authors, the implementation of the different peak matching methodologies did not translate to significant performance differences.

The identity of the peptides in the cross-linked di-peptide is derived from the highest scoring theoretical peptide in the theoretical library (Step 2, Figure 1.2). Programs such as IdentifyXLink in Pro-Crosslink, MassMatrix, MS2Assign, MS2Links, SearchXLink, X!Link, XLINK and xQUEST allow additional MS/MS verification of the peptides involved in the cross-link.

1.2.5 Identification of the cross-linked residues in the di-peptide

After identification of the peptides in the cross-linked dimer, the residue that is cross-linked must be identified (Step 3, Figure 1.2). In the case where only a single residue in each peptide is reactive toward the cross-linking reagent, the problem is trivial. However, if a greater number of cross-linkable residues are present in a di-peptide, the problem requires further analysis. Currently available software achieves this by comparison of the di-peptide product ion scan to fragment libraries, each generated according to preset fragmentation rules from the putative di-peptide in a particular cross-linker configuration (reviewed in (Paizs and Suhai, 2005a)). No program has implemented *de novo* sequencing as an approach, although xQUEST identifies uncross-linked peptides in this way.

1.2.5.1 Generating an MS/MS fragment library

Many programs (CrossSearch, Crux, GPMAW, MassMatrix, MS2Assign, MS2Links, MS2PRO, Pro-Crosslink, SearchXLinks, X!Link, XLINK and xQUEST) use defined models of peptide fragmentation (Barton *et al.*, 2007; Breci *et al.*, 2003; Huang *et al.*, 2005; Kapp *et al.*, 2003a; Khatun *et al.*, 2007b; Martin *et al.*, 2005a; Roepstorff and Fohlman, 1984a; Savitski *et al.*, 2007a; Savitski *et al.*, 2007b; Tabb *et al.*, 2003a; Zhang, 2004; Zhang, 2005) (reviewed in (Paizs and Suhai, 2005b; Papayannopoulos,

1995)) to perform *in silico* fragmentation of the identified di-peptide precursor ion. In this way the matched entries in the theoretical library of di-peptides are expanded to also include the product ion fragments for each entry. In the case of multiple possible cross-linked combinations for a given di-peptide, a different theoretical fragmentation spectrum is generated for each possibility. The fundamental chemistry behind peptide fragmentation is not yet fully understood, but a well-defined set of empirical fragmentation rules is known (Paizs and Suhai, 2005c). All programs utilize a fixed set of such rules, although MassMatrix allows the definition of additional custom fragmentation rules (Xu *et al.*, 2008b).

Most programs calculate all possible ions resulting from a single fragmentation on the backbone of putative di-peptide precursors, that is the a, b, c, x, y and z ion series (GPMAW, MS2Assign, MS2Links, Pro-Crosslink and SearchXLinks). In others the theoretical fragmentation library is constrained to the more abundant b and y product ions of these di-peptide precursors (Crux, MS2PRO, iXLINK, X!Link and xQUEST in ion-tag mode). Many programs also model a variety of additional fragment ion types. MS2Assign, MS2Links, MS2PRO and Pro-Crosslink include a search for the immonium ions for the amino acids H, M, W, Y, and F. GPMAW considers NH_3 and H_2O loss, whereas Crux, MS2Assign, MS2Links, Pro-Crosslink and doXLINK also include CO and, with the exception of Crux, CO_2 loss. Only SearchXLinks and PROWL's PeptideMap incorporate the proline effect, where breakage of the bond on the C-terminal side of P is typically not observed. Pro-Crosslink and X!Link model double fragmentation of a peptide, and the top-down methodology programs MS2Links and MS2PRO support a greater number of successive fragmentation steps. While CID fragmentation is the norm for MS3D software, SearchXLinks also incorporates in-source decay (ISD) and post-source decay (PSD), while MS2Links incorporates sustained off-resonance irradiation CID (SORI-CID).

1.2.5.2 Matching MS/MS spectra

Confirming the di-peptide identity and the cross-linking configuration that produced an experimental MS/MS spectrum involves the comparison of the experimental spectrum to the predicted spectra from a library of possible precursor ions (Step 3, Figure 1.2). However, predicted and observed spectra are seldom a perfect match. Many predicted products may be present in the experimental spectrum, but of such a low intensity as to go unobserved. Experimental spectra may also contain contaminants absent in the predicted spectrum.

To improve the efficiency of such comparisons, noise peaks can be excluded beforehand. While most instrument platforms are able to do this, some MS3D packages (MS2Assign, MS2Links, Pro-Crosslink and xQuest) incorporate pre-comparison noise filtering. xQUEST slides a window of *m/z* 1000 across each spectrum, including only the 250 most intense peaks within each window. The simplest method is the inclusion of peaks that are above an absolute (MS2Assign, Pro-Crosslink) or relative (MS2Links, Pro-Crosslink) intensity threshold. Though not automated, X-Links displays indicators of spectrum quality and allows users to manually exclude peaks. A number of stand-alone spectra filtering packages such as Decon2LS (Jaitly *et al.*, 2009) can also be used.

The method used to score the closeness-of-fit between the spectra is of critical importance to maximize true positive matches. A score can reflect a formal, statistical probability of a match with a certain confidence level (probabilistic), or it can be a value on an arbitrary, unbounded scale, with a minimum threshold required to qualify as an acceptable match (non-probabilistic). Some programs combine the two forms of scoring. MassMatrix calculates both a non-probabilistic and two probabilistic scores (Kapp *et al.*, 2003b; Tabb *et al.*, 2003b). xQUEST makes use of a non-probabilistic function that includes a probabilistic term (Rinner *et al.*, 2008b). Crux converts an initially non-probabilistic score into a probability estimate (McIlwain *et al.*, 2010c). The scores of true positive di-peptide assignments tend to be significantly lower than those of identified single peptides. In fact, the MassMatrix user manual recommends a probability threshold of approximately 0.2 for the assignment of di-peptides (Xu *et al.*, 2008a).

1.2.5.3 Non-probabilistic scoring

Non-probabilistic scoring of MS/MS spectrum matches is implemented in Crux, MS2Assign, MS2PRO, SearchXLinks, Pro-Crosslink, X-Links, X!Link and xQUEST. Most programs calculate these scores using simple scoring functions such as the number of theoretical fragments assigned (SearchXLinks), the number of experimental peaks successfully assigned (MS2Assign, SearchXLinks, X!Links), the percentage of peaks assigned (Pro-Crosslink), the number of peaks assigned, normalized to the number of amino acids in the precursor di-peptide (X!Link), or the sum of the intensities of all assigned peaks (SearchXLinks). Thus the score of MS2Assign, MS2PRO, Pro-Crosslink and X!Link is directly proportional to the number of assigned peaks. SearchXLinks uses each equation indicated above as well as similar equations for

specific fragments as terms in its scoring function. An additional term for the number of matches assigned to consecutive ions in the *b* or *y* fragment series can cause the SearchXLinks score to scale exponentially against the number of matches. The user can customize the scoring function by specifying a weighting for each term. X-Links uses the uniqueness of a mass within its theoretical library as a match score.

XQUEST uses an initial filter that considers only the single-chain b and y ions lacking a cross-linkable site, followed by a more complex scoring scheme that involves a cross correlation function (first introduced in SEQUEST (Yates, III *et al.*, 1995b)), the percentage of ion intensities in the matched spectrum that is contributed by matched peaks, and a term equal to the negative $\log_{10}$ of the probability of a random match. Crux uses a similar, normalized cross-correlation function.

1.2.5.4 Probabilistic scoring

Three distinct approaches have been implemented in iXLINK, MassMatrix and Crux to derive a score related to a statistical probability.

iXLINK employs a Bayesian scoring scheme based solely on established fragmentation principles that are implemented as nested probability functions (Zhang *et al.*, 2002b). These consider factors such as the consistency of deduced amino acid composition with observed immonium ions, the number and mass deviation (normalized to instrument accuracy) of matched b and y ions, the probability that unmatched peaks are noise, as well as the fraction of complementary (b-y pair) and contiguous (b_x and b_{x+1}) assignments in a fragmentation series.

MassMatrix performs hypothesis testing on MS/MS spectrum matches, based on a probability distribution for random matching, that can be binomial or non-parametric, which is estimated from the experimental data (Xu and Freitas, 2007). Crux presumes a Weibull distribution for the probability of a random match, and estimates a probability score by fitting the non-probabilistic score to this distribution (McIlwain *et al.*, 2010b).

1.2.6 Structure modelling

To date, no MS3D analysis software seamlessly integrates the identification of cross-linked peptides and residues with the verification or refinement of a structural model (Step 4, Figure 1.2). In many structural studies that reported MS analysis of chemically cross-linked proteins, distance constraints revealed by the position and reach of the cross-linker were entered into separate modelling programs such as VMD-XPLOR

(Schwieters and Clore, 2001), or were directly incorporated into subsequent homology modelling or docking studies (Jaitly *et al.*, 2009; Khatun *et al.*, 2007a; Singh *et al.*, 2008a; Yates, III *et al.*, 1995a; Zhang *et al.*, 2002a)**.** MSX-3D (Heymann *et al.*, 2008c), however, determines whether published structural models described in a PDB format (Berman *et al.*, 2003) are consistent with observed peptide cross-links and spatial reach of the chemical cross-linking reagent. MSX-3D also visualises the model and the spatial reach of the selected cross-linking reagent in a web applet.

1.2.7 Data input and output

In line with trends in the biological sciences towards high-throughput methodologies, MS3D studies also require the analysis of ever larger datasets. This is not supported in many of the older programs where data needs to be manually copied-and-pasted (CrossSearch, MS-Bridge and PeptideMap), or where only one spectrum can be uploaded at a time (ASAP, CLPM, MS2Assign, MSX-3D, Pro-Crosslink and GPMAW's ProteinXXX). CLPM, Crux, SearchXLinks, X!Link and XLINK have command line interfaces that allow the scripting of serial analyses in batch. However, with XLINK data files must be manually transferred into and out of a specific working folder for each analysis. Many of the newer MS3D packages, though, claim the handling of high data volume to be an explicit design goal (X!Link, X-Links and xQUEST). Specific data volumes, in terms of the number of spectra, have been demonstrated for Crux (3314 spectra) (McIlwain *et al.*, 2010a), X!Link (approximately 5000 spectra) (Lee *et al.*, 2007c) and xQUEST (3592 spectra) (Rinner *et al.*, 2008a). However, throughput bottlenecks can also arise with the server. In the case of xQUEST the authors found a variable upper limit to the size of the data file that could successfully be uploaded via the xQUEST web form.

1.2.8 Software release

Many MS3D software packages have been released as web-based services (CLPM, CrossSearch, MS2Assign, ProteinProspector's MS-Bridge, MSX-3D, PROWL's PeptideMapASAP, SearchXLinks and xQUEST) or as packages that can be downloaded (Crux, MassMatrix, Pro-Crosslink, X-Links, XLINK) or requested from the developer (CLPM, VIRTUALMSLAB, X-Link, X!Link, XLINK). The platform, dependencies and availability of the different software tools reviewed are listed in Table 1.3.

Table 1.3: Software format, availability of source code, dependencies and required operating system for software use, and web sites where software may be accessed.

Software	Ref[e]	Software availability	Open source[a]	Dependencies[b]	Platform[c]	URL[d]
ASAP	56	Free web service	No	None	NA	NW
CLPM	48	Free download	GNU-GPL	None	L, W	bioinformatics.ualr.edu/mbc/services/CLPM.html (source code: YxTang2@UALR.edu / minho_chae@yahoo.com)
CrossSearch	30	Free web service	No	None	NA	crosssearch.umkc.edu/prot_cross3/
Crux	29	Free download		None	W, L, Mac	noble.gs.washington.edu/proj/crux
FindLink		In-house (unreleased)	No	Unknown	NS	NW
Links + MS2Link	20	Free web service	No		NA	NW
MassMatrix	53	Proprietary, Free to academia	Proprietary, Free to academia	None	W	www.massmatrix.net
MS2Assign	39	Free web service	No	ASAP (see above)	NA	NW
MS2PRO	22	Free web service	No	None	NA	NW
MS-Bridge (in Protein Prospecter)	9	Free web service	Proprietary	None	NA	prospector.ucsf.edu
MSX-3D	14	Free web service	No	None	NA	proteomics-pbil.ibcp.fr
NIH-XL	64	In-house (unreleased)	No	Unknown	NS	NW
PeptideMap (in PROWL)	12	Free web service	No	None	NA	prowl.rockefeller.edu/
Pro-Crosslink	13	Free download	No		W	depts.washington.edu/ventures/UW_Technology/Express_Licenses/
GPMAW / ProteinXXX	61,62	GPMAW proprietary; ProteinXXX freeware	No	None	W	www.gpmaw.com
SearchXLinks	50	Free web service	No	None	W, L, U, S	NW
VIRTUALMS-LAB	11	Free download (on request)	Yes		W	NW (contact: ldk@science.uva.nl)
X!Link	23	Free download (on request)	NA	None	NS	NW (contact: yojlee@ucdavis.edu / yjlee@iastate.edu)
xComb	63	Free web service or download	Yes	Perl	NS	phenyx.proteomics.washington.edu/CXDB/index.cgi
X-Link	49	Free download (on request)	NA	None	NS	NW
XLINK (iXLINK & doXLINK)	43	Free download	NA	Perl, Java	C	tools.proteomecenter.org
X-Links	2	Free	NA	ICR-	W	NW

		download		2LS		
XQUEST	35	Free web service	NA	None	NA	www.xquest.org

[a] Availability of source code. NA=the files are not publicly available, but may be requested from the developers.
[b] The requirement of each software package for other installed packages, platforms or drivers
[c] Operating system platform. W=Windows, L=Linux, Mac=MacOS, U=Unix, S=Sun (versions not specified). NA=not applicable, NS=not specified, C=cross-platform.
[d] The website address where software packages may be accessed (verified on 15 Nov. 2010). NW=no website was specified, or the specified website is no longer available. In some cases the e-mail address of a contact person was supplied.
[e] Reference 64 is first cited in this table (Sinz and Wang, 2001).

1.3 Discussion

MS3D is a valuable addition to the arsenal of tools in structural biology. As an MS-based technique, MS3D is faster and less costly that traditional methods, and able to analyse samples at the nanogram level (Mann and Kelleher, 2008). The major advantage of MS3D is the size of molecular assemblies that are amenable to structural investigation. Developments in data analysis and cross-linking reagents are also broadening the scope of MS3D-type approaches and improving workflow efficiency. This is likely to increase the throughput of solved structures for mega-Dalton protein assemblies, complementing established techniques such as XRC and NMR.

To meet the computational demands of the technique, available software for MS3D analysis has continued to increase in sophistication. With an increased emphasis on high-throughput approaches, many of the more recent programs are designed to accommodate larger experimental datasets and generate larger theoretical libraries for comparison. The need for confident identification of low abundance cross-linked di-peptides within complex peptide mixtures, has been addressed with a more rigorous statistical treatment of the process of matching experimental against theoretical spectra. This allows low-abundance di-peptides to be detected within larger, more complex datasets with fewer false positives.

While a number of powerful analytical features have appeared in MS3D software, these are restricted to a few specific packages which tend to cater for very specific experimental methodologies. These typically involve different uses of isotope labelling and uncross-linked controls. However, speciality MS3D software also caters for protein-nucleic acid complexes (CLPM, MS2Links), top-down MS3D (MS2Links, MS2PRO), MALDI data (XLINK), ISD fragmentation (SearchXLinks) and PIR cross-linker methodology (X-Links).

GPMAW and VIRTUALMSLAB offer versatile and user-friendly graphic interfaces but do not have complete MS/MS-matching functionality. MSX-3D uniquely allows the verification of determined structures by MS3D but, lacking MS/MS analysis, should only be used to analyse simple experimental spectra. Pro-Crosslink, SearchXLinks and X!Link perform sound but rudimentary MS/MS analysis. For the best detection of cross-linked species Crux, xQUEST, XLINK and MassMatrix all implement sophisticated scoring schemes. Of these xQUEST and XLINK also exploit isotope labelling. Crux and XLINK can be run via scripts through a command line interface, and xQUEST is specially geared to process large datasets, but does demonstrate a slight assignment bias towards single-chain peptides. MassMatrix has the distinct advantage that its scoring scheme can be recalibrated by the user with new datasets.

Currently, an important deficiency is the absence of a single, integrated software package that also allows direct structural interpretation and modelling of protein complexes based on the spatial constrains revealed by the MS mapping data.

1.4 References

1. Ambler, R. P. and Meadway, R. J. (1968) The use of thermolysin in amino acid sequence determination. *Biochem.J.* **108,** 893-895.

2. Anderson, G. A., Tolic, N., Tang, X., Zheng, C., and Bruce, J. E. (2007c) Informatics strategies for large-scale novel cross-linking analysis. *J.Proteome.Res.* **6,** 3412-3421.

3. Anderson, G. A., Tolic, N., Tang, X., Zheng, C., and Bruce, J. E. (2007b) Informatics strategies for large-scale novel cross-linking analysis. *J.Proteome.Res.* **6,** 3412-3421.

4. Anderson, G. A., Tolic, N., Tang, X., Zheng, C., and Bruce, J. E. (2007a) Informatics strategies for large-scale novel cross-linking analysis. *J.Proteome.Res.* **6,** 3412-3421.

5. Back, J. W., Notenboom, V., de Koning, L. J., Muijsers, A. O., Sixma, T. K., de Koster, C. G., and de Jong, L. (2002) Identification of cross-linked peptides for protein interaction studies using mass spectrometry and ^{18}O labeling. *Anal.Chem.* **74,** 4417-4422.

6. Barton, S. J., Richardson, S., Perkins, D. N., Bellahn, I., Bryant, T. N., and Whittaker, J. C. (2007) Using statistical models to identify factors that have a role in defining the abundance of ions produced by tandem MS. *Anal.Chem.* **79,** 5601-5607.

7. Berman, H., Henrick, K., and Nakamura, H. (2003) Announcing the worldwide Protein Data Bank. *Nat.Struct.Biol.* **10,** 980.

8. Breci, L. A., Tabb, D. L., Yates, J. R., III, and Wysocki, V. H. (2003) Cleavage N-terminal to proline: analysis of a database of peptide tandem mass spectra. *Anal.Chem.* **75,** 1963-1971.

9. Breddam, K. and Meldal, M. (1992a) Substrate preferences of glutamic-acid-specific endopeptidases assessed by synthetic peptide substrates based on intramolecular fluorescence quenching. *Eur.J.Biochem.* **206,** 103-107.

10. Breddam, K. and Meldal, M. (1992b) Substrate preferences of glutamic-acid-specific endopeptidases assessed by synthetic peptide substrates based on intramolecular fluorescence quenching. *Eur.J.Biochem.* **206,** 103-107.

11. Chu, F., Baker, P. R., Burlingame, A. L., and Chalkley, R. J. (2010) Finding chimeras: a bioinformatics strategy for identification of cross-linked peptides. *Mol.Cell Proteomics.* **9,** 25-31.

12. Clauser, K. R., Baker, P., and Burlingame, A. L. (1999) Role of accurate mass measurement (+/- 10 ppm) in protein identification strategies employing MS or MS/MS and database searching. *Anal.Chem.* **71,** 2871-2882.

13. Craig, R. and Beavis, R. C. (2004) TANDEM: matching proteins with tandem mass spectra. *Bioinformatics.* **20,** 1466-1467.

14. de Koning, L. J., Kasper, P. T., Back, J. W., Nessen, M. A., Vanrobaeys, F., Van Beeumen, J., Gherardi, E., de Koster, C. G., and de Jong, L. (2006b) Computer-assisted mass spectrometric analysis of naturally occurring and artificially introduced cross-links in proteins and protein complexes. *FEBS J.* **273,** 281-291.

15. de Koning, L. J., Kasper, P. T., Back, J. W., Nessen, M. A., Vanrobaeys, F., Van Beeumen, J., Gherardi, E., de Koster, C. G., and de Jong, L. (2006a) Computer-assisted mass spectrometric analysis of naturally occurring and artificially introduced cross-links in proteins and protein complexes. *FEBS J.* **273,** 281-291.

16. Fenyo, D. (1997) A software tool for the analysis of mass spectrometric disulfide mapping experiments. *Comput.Appl.Biosci.* **13,** 617-618.

17. Gao, Q., Xue, S., Doneanu, C. E., Shaffer, S. A., Goodlett, D. R., and Nelson, S. D. (2006a) Pro-CrossLink. Software tool for protein cross-linking and mass spectrometry. *Anal.Chem.* **78,** 2145-2149.

18. Gao, Q., Xue, S., Doneanu, C. E., Shaffer, S. A., Goodlett, D. R., and Nelson, S. D. (2006b) Pro-CrossLink. Software tool for protein cross-linking and mass spectrometry. *Anal.Chem.* **78,** 2145-2149.

19. Heymann, M., Paramelle, D., Subra, G., Forest, E., Martinez, J., Geourjon, C., and Deleage, G. (2008a) MSX-3D: a tool to validate 3D protein models using mass spectrometry. *Bioinformatics.* **24,** 2782-2783.

20. Heymann, M., Paramelle, D., Subra, G., Forest, E., Martinez, J., Geourjon, C., and Deleage, G. (2008b) MSX-3D: a tool to validate 3D protein models using mass spectrometry. *Bioinformatics.* **24,** 2782-2783.

21. Heymann, M., Paramelle, D., Subra, G., Forest, E., Martinez, J., Geourjon, C., and Deleage, G. (2008c) MSX-3D: a tool to validate 3D protein models using mass spectrometry. *Bioinformatics.* **24,** 2782-2783.

22. Huang, Y., Triscari, J. M., Tseng, G. C., Pasa-Tolic, L., Lipton, M. S., Smith, R. D., and Wysocki, V. H. (2005) Statistical characterization of the charge state and residue dependence of low-energy CID peptide dissociation patterns. *Anal.Chem.* **77,** 5800-5813.

23. Jaitly, N., Mayampurath, A., Littlefield, K., Adkins, J. N., Anderson, G. A., and Smith, R. D. (2009) Decon2LS: An open-source software package for automated processing and visualization of high resolution mass spectrometry data. *BMC.Bioinformatics.* **10,** 87.

24. Jonic, S. and Venien-Bryan, C. (2009) Protein structure determination by electron cryo-microscopy. *Curr.Opin.Pharmacol.* **9,** 636-642.

25. Kang, S., Mou, L., Lanman, J., Velu, S., Brouillette, W. J., and Prevelige, P. E., Jr. (2009a) Synthesis of biotin-tagged chemical cross-linkers and their applications for mass spectrometry. *Rapid Commun.Mass Spectrom.* **23,** 1719-1726.

26. Kang, S., Mou, L., Lanman, J., Velu, S., Brouillette, W. J., and Prevelige, P. E., Jr. (2009b) Synthesis of biotin-tagged chemical cross-linkers and their applications for mass spectrometry. *Rapid Commun.Mass Spectrom.* **23,** 1719-1726.

27. Kang, S., Mou, L., Lanman, J., Velu, S., Brouillette, W. J., and Prevelige, P. E., Jr. (2009c) Synthesis of biotin-tagged chemical cross-linkers and their applications for mass spectrometry. *Rapid Commun.Mass Spectrom.* **23,** 1719-1726.

28. Kapp, E. A., Schutz, F., Reid, G. E., Eddes, J. S., Moritz, R. L., O'Hair, R. A., Speed, T. P., and Simpson, R. J. (2003a) Mining a tandem mass spectrometry database to determine the trends and global factors influencing peptide fragmentation. *Anal.Chem.* **75,** 6251-6264.

29. Kapp, E. A., Schutz, F., Reid, G. E., Eddes, J. S., Moritz, R. L., O'Hair, R. A., Speed, T. P., and Simpson, R. J. (2003b) Mining a tandem mass spectrometry database to determine the trends and global factors influencing peptide fragmentation. *Anal.Chem.* **75,** 6251-6264.

30. Kellersberger, K. A., Yu, E., Kruppa, G. H., Young, M. M., and Fabris, D. (2004) Top-down characterization of nucleic acids modified by structural probes using high-resolution tandem mass spectrometry and automated data interpretation. *Anal.Chem.* **76,** 2438-2445.

31. Khatun, J., Ramkissoon, K., and Giddings, M. C. (2007a) Fragmentation characteristics of collision-induced dissociation in MALDI TOF/TOF mass spectrometry. *Anal.Chem.* **79,** 3032-3040.

32. Khatun, J., Ramkissoon, K., and Giddings, M. C. (2007b) Fragmentation characteristics of collision-induced dissociation in MALDI TOF/TOF mass spectrometry. *Anal.Chem.* **79,** 3032-3040.

33. Khatun, J., Ramkissoon, K., and Giddings, M. C. (2007c) Fragmentation characteristics of collision-induced dissociation in MALDI TOF/TOF mass spectrometry. *Anal.Chem.* **79,** 3032-3040.

34. Kruppa, G. H., Schoeniger, J., and Young, M. M. (2003) A top down approach to protein structural studies using chemical cross-linking and Fourier transform mass spectrometry. *Rapid Commun.Mass Spectrom.* **17,** 155-162.

35. Lee, Y. J., Lackner, L. L., Nunnari, J. M., and Phinney, B. S. (2007c) Shotgun cross-linking analysis for studying quaternary and tertiary protein structures. *J.Proteome.Res.* **6,** 3908-3917.

36. Lee, Y. J., Lackner, L. L., Nunnari, J. M., and Phinney, B. S. (2007a) Shotgun cross-linking analysis for studying quaternary and tertiary protein structures. *J.Proteome.Res.* **6,** 3908-3917.

37. Lee, Y. J., Lackner, L. L., Nunnari, J. M., and Phinney, B. S. (2007b) Shotgun cross-linking analysis for studying quaternary and tertiary protein structures. *J.Proteome.Res.* **6,** 3908-3917.

38. Leiros, H. K., Brandsdal, B. O., Andersen, O. A., Os, V., Leiros, I., Helland, R., Otlewski, J., Willassen, N. P., and Smalas, A. O. (2004a) Trypsin specificity as elucidated by LIE calculations, X-ray structures, and association constant measurements. *Protein Sci.* **13,** 1056-1070.

39. Leiros, H. K., Brandsdal, B. O., Andersen, O. A., Os, V., Leiros, I., Helland, R., Otlewski, J., Willassen, N. P., and Smalas, A. O. (2004b) Trypsin specificity as elucidated by LIE calculations, X-ray structures, and association constant measurements. *Protein Sci.* **13,** 1056-1070.

40. Maiolica, A., Cittaro, D., Borsotti, D., Sennels, L., Ciferri, C., Tarricone, C., Musacchio, A., and Rappsilber, J. (2007) Structural analysis of multiprotein complexes by cross-linking, mass spectrometry, and database searching. *Mol.Cell Proteomics.* **6,** 2200-2211.

41. Mann, M. and Kelleher, N. L. (2008) Precision proteomics: the case for high resolution and high mass accuracy. *Proc.Natl.Acad.Sci.U.S.A* **105,** 18132-18138.

42. Martin, D. B., Eng, J. K., Nesvizhskii, A. I., Gemmill, A., and Aebersold, R. (2005b) Investigation of neutral loss during collision-induced dissociation of peptide ions. *Anal.Chem.* **77,** 4870-4882.

43. Martin, D. B., Eng, J. K., Nesvizhskii, A. I., Gemmill, A., and Aebersold, R. (2005a) Investigation of neutral loss during collision-induced dissociation of peptide ions. *Anal.Chem.* **77,** 4870-4882.

44. McHugh, L. and Arthur, J. W. (2008a) Computational methods for protein identification from mass spectrometry data. *PLoS Comput.Biol.* **4,** e12.

45. McHugh, L. and Arthur, J. W. (2008b) Computational methods for protein identification from mass spectrometry data. *PLoS Comput.Biol.* **4,** e12.

46. McIlwain, S., Draghicescu, P., Singh, P., Goodlett, D. R., and Noble, W. S. (2010e) Detecting cross-linked peptides by searching against a database of cross-linked peptide pairs. *J.Proteome.Res.* **9,** 2488-2495.

47. McIlwain, S., Draghicescu, P., Singh, P., Goodlett, D. R., and Noble, W. S. (2010c) Detecting cross-linked peptides by searching against a database of cross-linked peptide pairs. *J.Proteome.Res.* **9,** 2488-2495.

48. McIlwain, S., Draghicescu, P., Singh, P., Goodlett, D. R., and Noble, W. S. (2010d) Detecting cross-linked peptides by searching against a database of cross-linked peptide pairs. *J.Proteome.Res.* **9,** 2488-2495.

49. McIlwain, S., Draghicescu, P., Singh, P., Goodlett, D. R., and Noble, W. S. (2010b) Detecting cross-linked peptides by searching against a database of cross-linked peptide pairs. *J.Proteome.Res.* **9,** 2488-2495.

50. McIlwain, S., Draghicescu, P., Singh, P., Goodlett, D. R., and Noble, W. S. (2010a) Detecting cross-linked peptides by searching against a database of cross-linked peptide pairs. *J.Proteome.Res.* **9,** 2488-2495.

51. Nadeau, O. W., Wyckoff, G. J., Paschall, J. E., Artigues, A., Sage, J., Villar, M. T., and Carlson, G. M. (2008) CrossSearch, a user-friendly search engine for detecting chemically cross-linked peptides in conjugated proteins. *Mol.Cell Proteomics.* **7,** 739-749.

52. Nielsen, T., Thaysen-Andersen, M., Larsen, N., Jorgensen, F. S., Houen, G., and Hojrup, P. (2007) Determination of protein conformation by isotopically labelled cross-linking and dedicated software: Application to the chaperone, calreticulin. *International Journal of Mass Spectrometry* **268,** 217-226.

53. Ong, S. E., Blagoev, B., Kratchmarova, I., Kristensen, D. B., Steen, H., Pandey, A., and Mann, M. (2002b) Stable isotope labeling by amino acids in cell culture, SILAC, as a simple and accurate approach to expression proteomics. *Mol.Cell Proteomics.* **1,** 376-386.

54. Ong, S. E., Blagoev, B., Kratchmarova, I., Kristensen, D. B., Steen, H., Pandey, A., and Mann, M. (2002a) Stable isotope labeling by amino acids in cell culture,

SILAC, as a simple and accurate approach to expression proteomics. *Mol.Cell Proteomics.* **1,** 376-386.

55. Ong, S. E., Blagoev, B., Kratchmarova, I., Kristensen, D. B., Steen, H., Pandey, A., and Mann, M. (2002c) Stable isotope labeling by amino acids in cell culture, SILAC, as a simple and accurate approach to expression proteomics. *Mol.Cell Proteomics.* **1,** 376-386.

56. Paizs, B. and Suhai, S. (2005b) Fragmentation pathways of protonated peptides. *Mass Spectrom.Rev.* **24,** 508-548.

57. Paizs, B. and Suhai, S. (2005c) Fragmentation pathways of protonated peptides. *Mass Spectrom.Rev.* **24,** 508-548.

58. Paizs, B. and Suhai, S. (2005a) Fragmentation pathways of protonated peptides. *Mass Spectrom.Rev.* **24,** 508-548.

59. Panchaud, A., Singh, P., Shaffer, S. A., and Goodlett, D. R. (2010) xComb: a cross-linked peptide database approach to protein-protein interaction analysis. *J.Proteome Res.* **9,** 2508-2515.

60. Papayannopoulos, I. A. (1995) The interpretation of collision-induced dissociation tandem mass spectra of peptides. *Mass Spectrometry Reviews* **14,** 49-73.

61. Peri, S., Steen, H., and Pandey, A. (2001) GPMAW--a software tool for analyzing proteins and peptides. *Trends Biochem.Sci.* **26,** 687-689.

62. Perkins, D. N., Pappin, D. J., Creasy, D. M., and Cottrell, J. S. (1999) Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis* **20,** 3551-3567.

63. Rinner, O., Seebacher, J., Walzthoeni, T., Mueller, L. N., Beck, M., Schmidt, A., Mueller, M., and Aebersold, R. (2008a) Identification of cross-linked peptides from large sequence databases. *Nat.Methods* **5,** 315-318.

64. Rinner, O., Seebacher, J., Walzthoeni, T., Mueller, L. N., Beck, M., Schmidt, A., Mueller, M., and Aebersold, R. (2008b) Identification of cross-linked peptides from large sequence databases. *Nat.Methods* **5,** 315-318.

65. Rinner, O., Seebacher, J., Walzthoeni, T., Mueller, L. N., Beck, M., Schmidt, A., Mueller, M., and Aebersold, R. (2008c) Identification of cross-linked peptides from large sequence databases. *Nat.Methods* **5,** 315-318.

66. Rinner, O., Seebacher, J., Walzthoeni, T., Mueller, L. N., Beck, M., Schmidt, A., Mueller, M., and Aebersold, R. (2008d) Identification of cross-linked peptides from large sequence databases. *Nat.Methods* **5,** 315-318.

67. Roepstorff, P. and Fohlman, J. (1984a) Proposal for a common nomenclature for sequence ions in mass spectra of peptides. *Biomed.Mass Spectrom.* **11,** 601.

68. Roepstorff, P. and Fohlman, J. (1984b) Proposal for a common nomenclature for sequence ions in mass spectra of peptides. *Biomed.Mass Spectrom.* **11,** 601.

69. Savitski, M. M., Kjeldsen, F., Nielsen, M. L., Garbuzynskiy, S. O., Galzitskaya, O. V., Surin, A. K., and Zubarev, R. A. (2007a) Backbone carbonyl group basicities are related to gas-phase fragmentation of peptides and protein folding. *Angew.Chem.Int.Ed Engl.* **46,** 1481-1484.

70. Savitski, M. M., Kjeldsen, F., Nielsen, M. L., and Zubarev, R. A. (2007b) Relative specificities of water and ammonia losses from backbone fragments in collision-activated dissociation. *J.Proteome Res.* **6,** 2669-2673.

71. Schilling, B., Row, R. H., Gibson, B. W., Guo, X., and Young, M. M. (2003a) MS2Assign, automated assignment and nomenclature of tandem mass spectra of chemically crosslinked peptides. *J.Am.Soc.Mass Spectrom.* **14,** 834-850.

72. Schilling, B., Row, R. H., Gibson, B. W., Guo, X., and Young, M. M. (2003b) MS2Assign, automated assignment and nomenclature of tandem mass spectra of chemically crosslinked peptides. *J.Am.Soc.Mass Spectrom.* **14,** 834-850.

73. Schnaible, V., Wefing, S., Resemann, A., Suckau, D., Bucker, A., Wolf-Kummeth, S., and Hoffmann, D. (2002b) Screening for disulfide bonds in proteins by MALDI in-source decay and LIFT-TOF/TOF-MS. *Anal.Chem.* **74,** 4980-4988.

74. Schnaible, V., Wefing, S., Resemann, A., Suckau, D., Bucker, A., Wolf-Kummeth, S., and Hoffmann, D. (2002a) Screening for disulfide bonds in proteins by MALDI in-source decay and LIFT-TOF/TOF-MS. *Anal.Chem.* **74,** 4980-4988.

75. Schnolzer, M., Jedrzejewski, P., and Lehmann, W. D. (1996b) Protease-catalyzed incorporation of ^{18}O into peptide fragments and its application for protein sequencing by electrospray and matrix-assisted laser desorption/ionization mass spectrometry. *Electrophoresis* **17,** 945-953.

76. Schnolzer, M., Jedrzejewski, P., and Lehmann, W. D. (1996a) Protease-catalyzed incorporation of ^{18}O into peptide fragments and its application for protein sequencing by electrospray and matrix-assisted laser desorption/ionization mass spectrometry. *Electrophoresis* **17,** 945-953.

77. Schwieters, C. D. and Clore, G. M. (2001) The VMD-XPLOR visualization package for NMR structure refinement. *J.Magn Reson.* **149,** 239-244.

78. Seebacher, J., Mallick, P., Zhang, N., Eddes, J. S., Aebersold, R., and Gelb, M. H. (2006a) Protein cross-linking analysis using mass spectrometry, isotope-coded cross-linkers, and integrated computational data processing. *J.Proteome.Res.* **5,** 2270-2282.

79. Seebacher, J., Mallick, P., Zhang, N., Eddes, J. S., Aebersold, R., and Gelb, M. H. (2006b) Protein cross-linking analysis using mass spectrometry, isotope-coded cross-linkers, and integrated computational data processing. *J.Proteome.Res.* **5,** 2270-2282.

80. Singh, P., Shaffer, S. A., Scherl, A., Holman, C., Pfuetzner, R. A., Larson Freeman, T. J., Miller, S. I., Hernandez, P., Appel, R. D., and Goodlett, D. R. (2008a) Characterization of protein cross-links via mass spectrometry and an open-modification search strategy. *Anal.Chem.* **80,** 8799-8806.

81. Singh, P., Shaffer, S. A., Scherl, A., Holman, C., Pfuetzner, R. A., Larson Freeman, T. J., Miller, S. I., Hernandez, P., Appel, R. D., and Goodlett, D. R. (2008c) Characterization of protein cross-links via mass spectrometry and an open-modification search strategy. *Anal.Chem.* **80,** 8799-8806.

82. Singh, P., Shaffer, S. A., Scherl, A., Holman, C., Pfuetzner, R. A., Larson Freeman, T. J., Miller, S. I., Hernandez, P., Appel, R. D., and Goodlett, D. R. (2008b) Characterization of protein cross-links via mass spectrometry and an open-modification search strategy. *Anal.Chem.* **80,** 8799-8806.

83. Sinz, A. and Wang, K. (2001) Mapping protein interfaces with a fluorogenic cross-linker and mass spectrometry: application to nebulin-calmodulin complexes. *Biochemistry* **40,** 7903-7913.

84. Tabb, D. L., Smith, L. L., Breci, L. A., Wysocki, V. H., Lin, D., and Yates, J. R., III (2003b) Statistical characterization of ion trap tandem mass spectra from doubly charged tryptic peptides. *Anal.Chem.* **75,** 1155-1163.

85. Tabb, D. L., Smith, L. L., Breci, L. A., Wysocki, V. H., Lin, D., and Yates, J. R., III (2003a) Statistical characterization of ion trap tandem mass spectra from doubly charged tryptic peptides. *Anal.Chem.* **75,** 1155-1163.

86. Takao, T., Hori, H., Okamoto, K., Harada, A., Kamachi, M., and Shimonishi, Y. (1991b) Facile assignment of sequence ions of a peptide labelled with ^{18}O at the carboxyl terminus. *Rapid Commun.Mass Spectrom.* **5,** 312-315.

87. Takao, T., Hori, H., Okamoto, K., Harada, A., Kamachi, M., and Shimonishi, Y. (1991a) Facile assignment of sequence ions of a peptide labelled with ^{18}O at the carboxyl terminus. *Rapid Commun.Mass Spectrom.* **5,** 312-315.

88. Tang, X., Munske, G. R., Siems, W. F., and Bruce, J. E. (2005b) Mass spectrometry identifiable cross-linking strategy for studying protein-protein interactions. *Anal.Chem.* **77,** 311-318.

89. Tang, X., Munske, G. R., Siems, W. F., and Bruce, J. E. (2005a) Mass spectrometry identifiable cross-linking strategy for studying protein-protein interactions. *Anal.Chem.* **77,** 311-318.

90. Tang, Y., Chen, Y., Lichti, C. F., Hall, R. A., Raney, K. D., and Jennings, S. F. (2005c) CLPM: a cross-linked peptide mapping algorithm for mass spectrometric analysis. *BMC.Bioinformatics.* **6 Suppl 2,** S9.

91. Tang, Y., Chen, Y., Lichti, C. F., Hall, R. A., Raney, K. D., and Jennings, S. F. (2005d) CLPM: a cross-linked peptide mapping algorithm for mass spectrometric analysis. *BMC.Bioinformatics.* **6 Suppl 2,** S9.

92. Tang, Y., Chen, Y., Lichti, C. F., Hall, R. A., Raney, K. D., and Jennings, S. F. (2005e) CLPM: a cross-linked peptide mapping algorithm for mass spectrometric analysis. *BMC.Bioinformatics.* **6 Suppl 2,** S9.

93. Taverner, T., Hall, N. E., O'Hair, R. A., and Simpson, R. J. (2002b) Characterization of an antagonist interleukin-6 dimer by stable isotope labeling, cross-linking, and mass spectrometry. *J.Biol.Chem.* **277,** 46487-46492.

94. Taverner, T., Hall, N. E., O'Hair, R. A., and Simpson, R. J. (2002a) Characterization of an antagonist interleukin-6 dimer by stable isotope labeling, cross-linking, and mass spectrometry. *J.Biol.Chem.* **277,** 46487-46492.

95. Wefing, S., Schnaible, V., and Hoffmann, D. (2006c) SearchXLinks. A program for the identification of disulfide bonds in proteins from mass spectra. *Anal.Chem.* **78,** 1235-1241.

96. Wefing, S., Schnaible, V., and Hoffmann, D. (2006b) SearchXLinks. A program for the identification of disulfide bonds in proteins from mass spectra. *Anal.Chem.* **78,** 1235-1241.

97. Wefing, S., Schnaible, V., and Hoffmann, D. (2006a) SearchXLinks. A program for the identification of disulfide bonds in proteins from mass spectra. *Anal.Chem.* **78,** 1235-1241.

98. Wong, S. S. *Chemistry of protein conjugation and cross-linking*, 1 Ed., CRC-Press, Washington, 2010.

99. Xu, H. and Freitas, M. A. (2007) A mass accuracy sensitive probability based scoring algorithm for database searching of tandem mass spectrometry data. *BMC.Bioinformatics.* **8,** 133-142.

100. Xu, H., Zhang, L., and Freitas, M. A. (2008a) Identification and characterization of disulfide bonds in proteins and peptides from tandem MS data by use of the MassMatrix MS/MS search engine. *J.Proteome.Res.* **7,** 138-144.

101. Xu, H., Zhang, L., and Freitas, M. A. (2008b) Identification and characterization of disulfide bonds in proteins and peptides from tandem MS data by use of the MassMatrix MS/MS search engine. *J.Proteome.Res.* **7,** 138-144.

102. Xu, H., Zhang, L., and Freitas, M. A. (2008c) Identification and characterization of disulfide bonds in proteins and peptides from tandem MS data by use of the MassMatrix MS/MS search engine. *J.Proteome.Res.* **7,** 138-144.

103. Yates, J. R., III, Eng, J. K., McCormack, A. L., and Schieltz, D. (1995a) Method to correlate tandem mass spectra of modified peptides to amino acid sequences in the protein database. *Anal.Chem.* **67,** 1426-1436.

104. Yates, J. R., III, Eng, J. K., McCormack, A. L., and Schieltz, D. (1995b) Method to correlate tandem mass spectra of modified peptides to amino acid sequences in the protein database. *Anal.Chem.* **67,** 1426-1436.

105. Ye, X., Luke, B., Andresson, T., and Blonder, J. (2009) ^{18}O stable isotope labeling in MS-based proteomics. *Brief.Funct.Genomic.Proteomic.* **8,** 136-144.

106. Young, M. M., Tang, N., Hempel, J. C., Oshiro, C. M., Taylor, E. W., Kuntz, I. D., Gibson, B. W., and Dollinger, G. (2000) High throughput protein fold identification by using experimental constraints derived from intramolecular cross-links and mass spectrometry. *Proc.Natl.Acad.Sci.U.S.A* **97,** 5802-5806.

107. Zhang, H., Tang, X., Munske, G. R., Tolic, N., Anderson, G. A., and Bruce, J. E. (2009) Identification of protein-protein interactions and topologies in living cells with chemical cross-linking and mass spectrometry. *Mol.Cell Proteomics.* **8,** 409-420.

108. Zhang, N., Aebersold, R., and Schwikowski, B. (2002a) ProbID: a probabilistic algorithm to identify peptides through sequence database searching using tandem mass spectral data. *Proteomics.* **2,** 1406-1412.

109. Zhang, N., Aebersold, R., and Schwikowski, B. (2002b) ProbID: a probabilistic algorithm to identify peptides through sequence database searching using tandem mass spectral data. *Proteomics.* **2,** 1406-1412.

110. Zhang, Z. (2004) Prediction of low-energy collision-induced dissociation spectra of peptides. *Anal.Chem.* **76,** 3908-3922.

111. Zhang, Z. (2005) Prediction of low-energy collision-induced dissociation spectra of peptides with three or more charges. *Anal.Chem.* **77,** 6364-6373.

Chapter 2

The development of AnchorMS, a bioinformatics tool to elucidate the structure of protein complexes by analysis of mass spectra of chemically cross-linked complexes.

2.1. Introduction

The structural elucidation of biological molecules plays a critical role in the investigation of their in vivo function. Elucidated structural features can often be related to specific functions through distinct mechanical, chemical or immunological properties. The traditional methods of X-ray crystallography (XRC) and nuclear magnetic resonance (NMR) were successfully employed in the determination of high-resolution structures for single proteins. However, multi-protein complexes such as SAGA (Lee et al., 2011) are in the mega Dalton range, exceeding the molecular weight range suitable for XRC and NMR. Cryo-electron microscopy (cryo-EM) has been applied to the structural investigation of mega Dalton protein complexes, but is not yet capable of a resolution suitable for mechanistic studies. Furthermore, there remains a range of protein complex sizes between the upper Dalton limit of XCR and NMR and the lower Dalton limit of cryo-EM for which suitable methods need to be developed.

To complement traditional methods, mass spectrometry has been applied to enzymatic digestions of chemically cross-linked protein complexes in order to identify cross-linked residues. The presence of cross-linked residues places constraints on the possible distances between those residues within the native conformation of the protein complex. By identifying the cross-linked residues and the implied distance constraints, the orientation and contact points of sub-units of known structure within the larger protein complex can more accurately be determined. This methodology is referred to as ‘MS3D’ and shows promise as a complementary method to existing techniques. The development of MS3D as a reliable technique requires the creation of appropriate software with functionality beyond that of single-peptide identification packages.

The most challenging component of MS3D analysis is the identification of cross-linked di-peptides. Because cross-linking reactions are typically low-yield, di-peptides form a minor component within sample mixtures that are often complex.

MS3D software generates theoretical libraries of possible di-peptides as well as

expected precursor MS^1 spectra and, in many cases, fragment MS^2 spectra. These expected spectra are compared to the observed MS^1 and MS^2 spectra in order to identify precursors likely to be cross-linked di-peptides. Both of these tasks are computationally demanding where sample mixtures are complex. Many software packages ameliorate this by limiting the size of the theoretical library according to their own criteria. For example, Crux, MS2PRO, XLINK and xQUEST only consider the more abundant *b* and *y* fragment ions.

A major difference between software packages is how partial matches between observed and expected spectra are evaluated and scored. Less sophisticated programmes merely count matching fragment peaks. A few has gone one step further by normalizing against the size of the analyte di-peptide. More sophisticated software makes use of cross-correlation first introduced in SEQUEST (Yates, III *et al.*, 1995). Other tools, such as XLINK, MassMatrix and Crux, apply pre-defined probability distributions to preliminary score values in order to convert a score value into the probability of a true positive match.

The score value baseline or the probability distribution can be calibrated against an experimental dataset in order to bring it in line with real world trends. However, the choice of datasets can bias the behaviour of the software. For example, xQuest demonstrates a bias towards assigning MS^2 spectra as single-stranded peptides rather than di-peptides.

Even where scoring schemes are sophisticated, many of the available MS3D programmes are designed around very specific, specialized experimental approaches such as the “MIX” heavy isotope labelling procedure upon which the X-Link package depends (Taverner *et al.*, 2002). A minority of available packages remain applicable to general MS3D experimental designs without relying on specialized sample treatments.

We present AnchorMS as an open-source tool for identifying cross-linked di-peptides from the data generated by MS^1 and MS^2 analysis of digested proteins following chemical cross-linking, as part of an MS3D workflow (Mayne and Patterton, 2011; Sinz, 2003). AnchorMS utilizes a novel scoring scheme that makes use of a baseline based on unbiased datasets from simulations of false positive matching of randomly generated di-peptides. AnchorMS also uniquely considers the co-fragmentation of di-peptides of identical mass that differ only in which residues are cross-linked. In addition to calculating a primary score for detecting false positive matches (the *N*

Score), AnchorMS also calculates a unique and specialized secondary score for distinguishing between sequence-identical candidate assignments (the *UMC Score*).

AnchorMS can be used locally via the command line, and is also a freely accessible web application with an interface coded in PHP/XHTML. The free web service is hosted on a Linux server affiliated with the University of the Free State at cbio.ufs.ac.za/AnchorMS. All source code is licensed as open-source and is available on request. The core functionalities of AnchorMS are implemented in a series of Python scripts, which collectively form an importable Python package.

Based on supplied protein sequences, protease specificities and cross-linking reagent information, AnchorMS generates libraries of expected precursors and fragments in the form of theoretical MS^1 and MS^2 spectra. Experimental and theoretical spectra are compared to identify matching peaks. Overall, matching of MS^2 spectra is scored by a non-probabilistic method and uses the simulation derived level of false positive matching as a reference baseline. A ranked list of di-peptide precursors detected in the experimental spectra along with scores is displayed and written to file.

The functionality of AnchorMS and its underlying source code are described and discussed in detail below. We begin by outlining the overarching workflow and structure of the AnchorMS programme. We then follow, step-by-step, each salient computational task performed by AnchorMS, highlighting noteworthy software features and examining aspects of the code implementing each task.

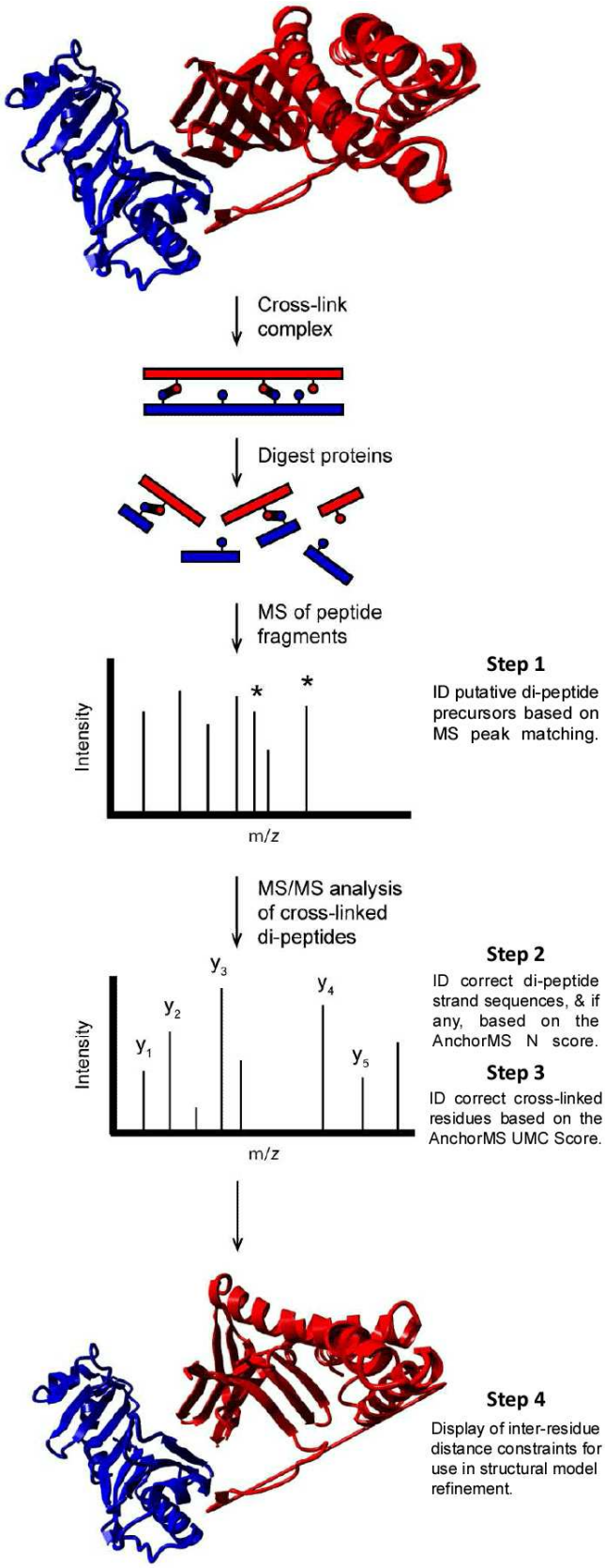


Figure 2.1: The illustration shows the generalized workflow for MS3D software analysis and indicates how AnchorMS addresses each step.

2.2. Workflow and organisation

The structural elucidation of multi-subunit protein complexes by identifying chemically cross-linked residues involves several experimental steps. After selection of the cross-linking reagent and possible affinity purification, the cross-linked peptides must be identified. As a second step, the cross-linked residues must be identified. Depending on the speed at which a mass spectrometer can perform MS^1 followed by MS^2, the two steps may be performed as part of a single run, or alternatively as two separate runs. From the MS analysis, precursors of interest can be identified. Where MS^2 is performed in a separate run, the precursors of interest can be analysed exclusively.

After the MS^1 and MS^2 experimental analyses are complete, the resulting data file must be uploaded to AnchorMS. In order to interpret the uploaded mass spectra, AnchorMS requires information about the experimental details of the upstream treatment of the analyte sample. Experimental details supplied by the user include proteases, cross-linking reagents, protein sequences and instrument mass accuracy. This information is captured by the AnchorMS web interface as a text file (User_Input.py), formatted as Python code, and will be read from the text file when the information is required for AnchorMS calculations. AnchorMS will perform the necessary calculations and display the results through the same web interface.

The AnchorMS package is made up by two distinct parts created to function as independent components within AnchorMS: the web-based front end and the computational back end. The results of the back end analysis are written to file (Results.txt) where it is accessed by the front end.

The web-based front end of AnchorMS is coded in PHP/HTML and implemented as a set of several scripts. A web form (AMSwebform.php) and a processing script (Processing.php) capture all user input. Processing.php places the information pertaining to each job request into a separate folder on the server, and executes the Python back end. Results generated by the Python back end are displayed in the front end by Results.php. Upon conclusion of analysis, the Python back end initiates a PHP script (email_user.php) that emails to the user a link to a web page displaying the results for later reference. These results can also be downloaded as a text file.

The back end is coded in Python due to its widespread use in the bioinformatics field and the availability of many libraries for biological applications. The back end is

comprised of a number of separate Python modules, each of which houses functionality for a set of specific, related computational tasks. Figure 2.2 summarizes the organisation of the back end Python modules that perform all of the computational tasks and outlines the dependency hierarchy of the modules, indicating where in the back end code elements of each module are called. Figure 2.3 follows the flow of information in an AnchorMS analysis, and details the computational task performed at each step of the process in the order in which they are performed. In describing the organisation and functioning of these Python modules, we walk through a typical AnchorMS analysis.

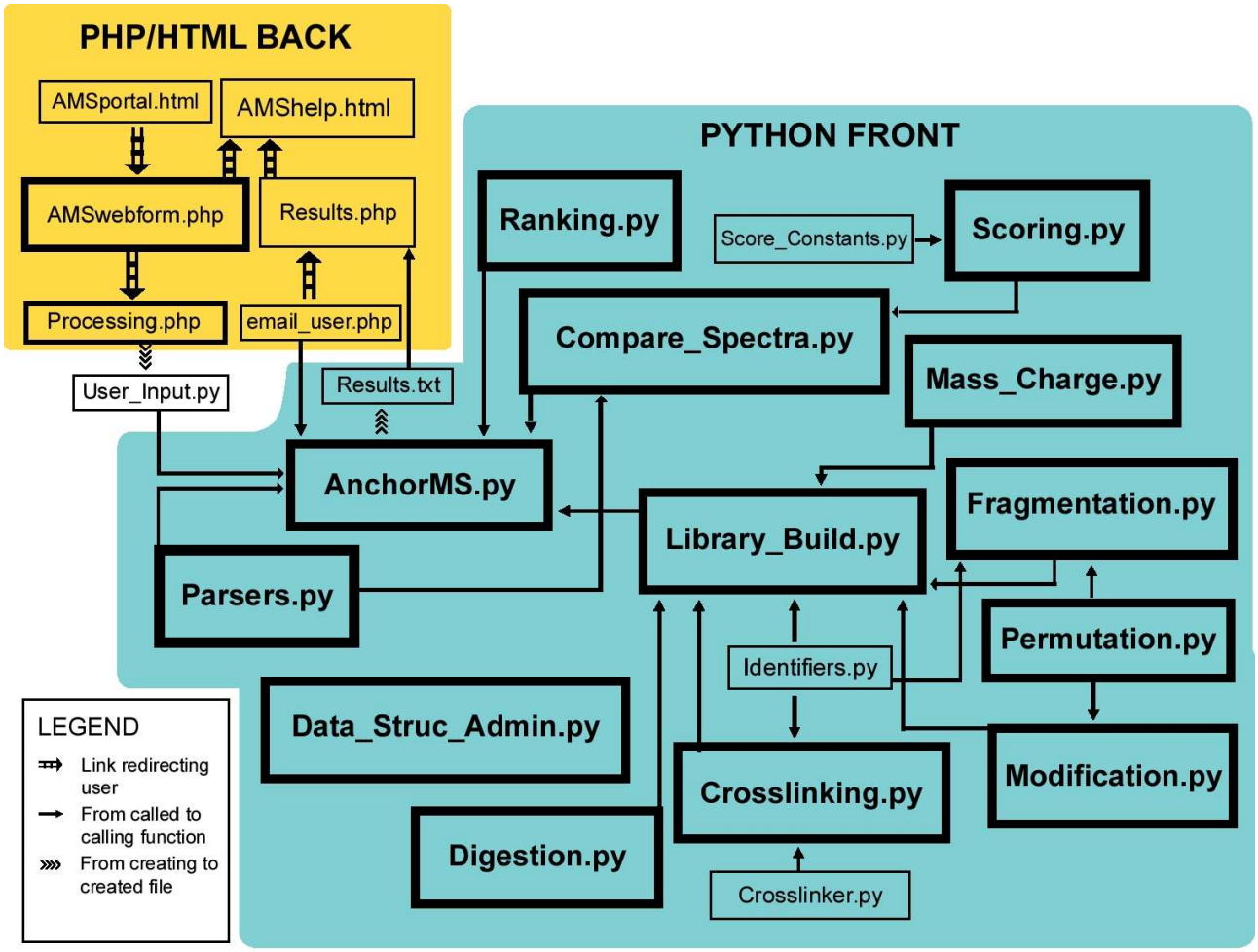


Figure 2.2: A diagram showing the organization of code within AnchorMS. Boxes indicate distinct modules housed in separate files. Background shading groups the elements of the web front end and Python back end. Arrows leading from functions or objects point to code units that call or instantiate them. Double-line arrows indicate the order of viewing by the user. Chevron arrows denote files that are created by specific code elements.

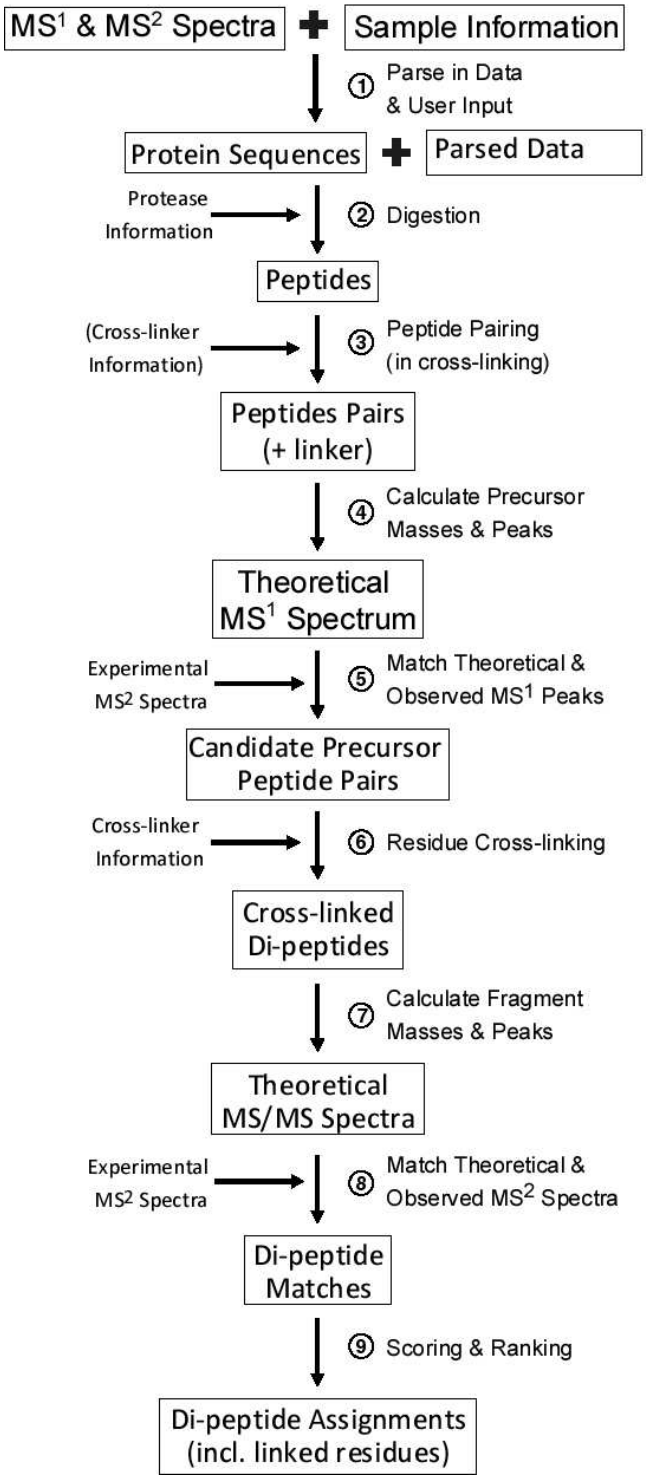


Figure 2.3: The flow of information between various software modules constitutes the digital workflow of AnchorMS. AnchorMS analyses mass spectrometric data from digested protein samples according to the workflow on the left, in order to identify di-peptide species present. This *in silico* workflow of AnchorMS is similar to the experimental workflow employed to prepare samples for mass spectrometric analysis in an MS3D experiment. 1) Uploaded MS data and user input regarding experimental conditions during sample preparation are parsed into AnchorMS. 2) Known protein sequences are digested *in silico* using the cleavage specificity of the protease used in sample preparation, including potential missed cleavages, thus calculating the sequences of all possible resultant peptide molecules. 3)

The dual sequences of each possible cross-linked peptide dimer are calculated by pairing peptide sequences respectively containing residues reactive to each of the functional groups of the "cross-linked" cross-linking agent employed. 4) The mass and *m/z* peak value of each di-peptide including the chemical cross-link is calculated to generate a theoretical library of possible di-peptide precursors detectable in an MS^1 spectrum. 5) This theoretical library is matched against the observed MS^1 spectrum, identifying peaks representing putative di-peptides in the sample. 6) All possible cross-linking sites within each putative di-peptide precursor are calculated to generate a theoretical precursor library of di-peptide molecules present. 7) Each di-peptide molecule in theoretical precursor library is *in silico* "CID fragmented" to generate a theoretical MS^2 spectrum for each possible *b* and *y* ion. 8) The observed MS^2 spectra of putative di-peptide precursors are matched against these theoretical MS^2 spectra to confirm the presence of putative di-peptides in the sample. 9) MS^2 spectrum matches are scored and ranked on the basis of score value.

A back end analysis is initiated by executing the AnchorMS.py module. AnchorMS.py co-ordinates the component elements of an AnchorMS analysis by calling functions from the other modules, as required.

In summary, AnchorMS.py oversees parsing of experimental data, precursor and fragment library construction, MS^1 and MS^2 spectrum matching, match scoring and candidate ranking. To perform each of these functionalities AnchorMS.py calls functions from the modules Parsers.py, Library_Build.py, Compare_Spectra.py, Scoring.py and Ranking.py respectively. Library_Build.py, in turn, oversees the construction of both precursor and fragment libraries making use of the modules Digestion.py, Crosslinking.py, Modifications.py, Fragmentation.py, Permutations.py and Identifiers.py.

Upon initiation, AnchorMS.py uses the module Parsers.py to parse the experimental information as well as the mass spectrometry data file into the AnchorMS back end. In this way Parsers.py ensures that AnchorMS is compatible with a wide variety of mass spectra file formats.

Next, AnchorMS.py uses the Library_Build.py module to construct a theoretical library of possible di-peptide precursors along with a theoretical MS^1 spectrum populated by expected precursor peaks. To do this, Library_Build.py performs *in silico* digestion and cross-linking using the modules Digestion.py and Crosslinking.py, respectively. Library_Build.py also simulates the addition of possible induced or post-translational modifications (PTMs) using the Modifications.py module. For the task of defining the complete combinatorial space of possible modifications, Modifications.py makes use of Permutations.py.

The theoretical, expected precursor mass spectrum constructed by Library_Build.py is matched by AnchorMS.py against the observed precursor mass spectrum using the module Compare_Spectra.py. This round of matching detects potential di-peptides in the observed precursor mass spectrum.

For each putative di-peptide precursor, AnchorMS uses Library_Build.py to generate a theoretical tandem MS spectrum (MS^2), populated by expected fragment ions that would result from the CID fragmentation of the di-peptide precursor. To build this library of theoretical MS^2 spectra, Library_Build.py uses Crosslinking.py to determine all possible cross-linked residue sites in the di-peptide precursor, and Fragmentation.py to perform *in silico* collision induced dissociation (CID) fragmentation of the di-peptide precursor for each cross-linked residue site configuration.

Using the module Compare_Spectra.py, AnchorMS.py matches each theoretical MS^2 spectrum of expected fragments against the observed MS^2 spectrum to confirm the di-peptide precursor identity, and determine which residues were cross-linked. AnchorMS.py calculates the quality of each match using the module Scoring.py. Scoring.py defines the AnchorMS scoring scheme, but imports from a separate module, Score_Constants.py, those constant values which have been determined through calibration against data from a false positive matching simulation (see Chapter 3). AnchorMS employs the Ranking.py module to rank all the putative assignments for each observed MS^2 spectrum according to the calculated score values.

As a final output, a list of observed di-peptides is displayed by the front end in appropriately formatted HTML and downloadable as a text file. Along with each di-peptide, the inferred distance constraint as well as a set of score values reflecting the quality of the spectrum match is also provided.

In conjunction with the known subunit structures, these distance constraints can be used to refine the structure of the multi-subunit protein complex using third party software such as VMD-XPLOR (Schwieters and Clore, 2001).

Following on from this workflow overview, this chapter will describe each general functionality and module within AnchorMS, and discuss algorithms implemented in each. The chapter will follow the AnchorMS workflow depicted in Figure 2.3 above, with each successive section detailing a computational step in that schema.

2.3. Parsing and data formats

2.3.1 Data formats commonly used for MS data

Experimental mass spectra can be uploaded in a number of formats. Mass spectrometers produce data files in propriety, binary formats (such as .asc, .pkl, .raw and .wiff files), which are specific to each instrument manufacturer and generally only directly viewable using proprietary software (see Table 2.1, below). However, from within such proprietary software packages, data can be exported into a text based format that is readable by a package capable of processing ASCII encoded text.

Text-based data files may be formatted as flat text or in Extensible Markup Language (XML). The .ms2 (McDonald *et al.*, 2004) and .mgf formats are among the most utilized flat text formats, and were first introduced in association with the SEQUEST (Yates, III *et al.*, 1995) and MASCOT (Perkins *et al.*, 1999) packages, respectively. Table 2.2 below shows example lines from both of these widely used flat text MS data formats.

XML is a general file format (not specific to MS), where information content is organised and grouped into categories or sections and sub-categories or sub-sections by ‘tags’, which parenthesize the information content. XML ‘tags’ are identified in the text as being situated between the characters “<” and “>”, and contain category labels and meta-information. XML is well suited for housing metadata, and for organizing data in a general way because individual data elements are labelled. For this reason, XML-structured formats are increasingly favoured when new data formats or format standards are proposed (Stromback *et al.*, 2007).

In the past a number of XML formats for MS data have been used. There have since been several attempts to standardize the format for MS data files. Amongst these, the mzData (Orchard *et al.*, 2005) and mzXML (Pedrioli *et al.*, 2004) file formats have gained widespread use, each originally proposed as a XML-structured format standard. More recently, the mzML format (formerly named dataXML) was developed as a combination of the mzData and mzXML formats. While other formats are still used, the mzML format currently serves as the primary standard for MS data (Deutsch, 2008; Deutsch, 2010; Martens *et al.*, 2011). For a data file to conform to either of these three XML formats, it must contain a large number of metadata fields. In all cases each MS run, each round of data processing, each scan and each peak are stored in distinct and separate tags.

2.3.2 Data file formats supported by AnchorMS

Currently, only widely used MS data file formats are catered for. Table 2.1 lists a variety of widely used MS data file formats and indicates which are supported by AnchorMS, including raw MS data file formats associated with several MS instrument manufacturers and commercial software packages. AnchorMS supports experimental spectra in flat text (.ms2 and .mgf), in XML-structured formats (.mzData, .mzXML and mzML), and in several proprietary, binary formats (.d, .raw and .wiff).

Instrument manufacturer	Associated software	MS data file extension	Accessibility	File type	AnchorMS supported
Bruker Daltronics	NA	.dtd, .xsd, .baf, .yep	Proprietary	Binary	NO
Hitachi	NA	.dat, .dad, .msd	Proprietary	Binary	NO
LECO	NA	.smp	Proprietary	Binary	NO
Micromass	MassLynx	.pkl	Proprietary	Binary	NO
Shimadzu	GCMSSolution	.qgd, .lcd, .qld	Proprietary	Binary	NO
Varian	NA	.sms, .xms	Proprietary	Binary	NO
Thermo Scientific / Waters / PerkinElmer	XCalibur / MassLynx / TurboMass	.raw	Proprietary	Binary	YES
ABI/Sciex	Analyst	.wiff	Proprietary	Binary	YES
Agilent	MassHunter	.d	Proprietary	Binary	YES
Finnigan	XCalibur	.asc, .dta	Open	Flat text	NO
Thermo Scientific	Grams	.pks	Open	Flat text	NO
	SEQUEST	.ms2	Open	Flat text	YES
NA	MASCOT	.mgf	Open	Flat text	YES
NA	NA	.mzData	Open	XML text	YES
NA	NA	.mzXML	Open	XML text	YES
NA	NA	.mzML	Open	XML text	YES
NA = Not applicable					

Table 2.1: The proprietary MS data file types for several MS instrument manufacturers.

Data format		Information structure in file	Notes
.ms2	Rubric	S <1st scan #> <2nd scan #> <precursor *m/z*> Z <charge> <precursor mass> <fragment *m/z*> <fragment intensity> ..	A scan ("S") line may map to multiple MS^2 spectra, each differing in charge state.
	Example	S 001267 001267 2400 Z 4 601.007823 275.96 585.5740000078350 ..	
.mgf	Rubric	BEGIN IONS TITLE=<Any text: description> CHARGE=<charge>+ PEPMASS=<Precursor mass> <fragment *m/z*> <fragment intensity> .. END IONS	The "Title" line may contain any text / description.
	Example	BEGIN IONS TITLE= SCAN1370-001370__[2493.38,4+] CHARGE=4+ 160.31 6530.9992394992 .. END IONS	

Table 2.2: Rubric and example entries for a single MS2 spectrum in several flat text MS data file formats.

2.3.3 Parsing of data files in AnchorMS and module organization

Within AnchorMS, the Parsers.py module is responsible for parsing in experimental mass spectra (see Figure 2.4, below). For each of the text-based file formats supported by AnchorMS, the Parsers.py module contains a dedicated function to specifically parse in each of the text-based file formats supported by AnchorMS. Thus, the functions *ParseIn_ms2()*, *ParseIn_mgf()*, *ParseIn_mzData()*, *ParseIn_mzXML()* and *ParseIn_mzML()* parse in .ms2, .mgf, .mzData, .mzXML and .mzML format data, respectively.

Proprietary, binary file formats are first converted to a .mzML file by the function

convert_propfile(), using the external utility *MSConvert.exe* (Kessner *et al.*, 2008). *MSConvert* was initially implemented as part of the suite of bioinformatic tools, *ProteoWizard*, but has since been released as a free, separate application. *MSConvert.exe* reads the proprietary binary file and creates an equivalent .mzML file. This .mzML file is then parsed in by the *ParseIn_mzML()* function.

Within the Parsers.py module, the *parser_delegate()* function calls the appropriate function to parse in the uploaded experimental MS data, based on the file extension. Therefore, AnchorMS does not validate the file format; the uploaded file must be correctly named. However, *parser_delegate()* does check the file name of the uploaded file to see if its file extension is one that is supported by AnchorMS. An error message is displayed if it is not.

Internally, AnchorMS stores each mass spectrum as a Python list of *m/z* peak values. The current version of AnchorMS does not make use of intensity information. However, future versions of AnchorMS may include peak intensity information in the analysis. Multiple MS^2 spectra are nested within a Python dictionary, each mapping to its MS^1 *m/z* peak value. Therefore, the uploaded MS^2 dataset is stored as a dictionary of lists, with each list containing the fragment peak values of a particular experimental MS^2 spectrum. Further meta-data that may be stored in MS data files is not currently parsed or utilized.

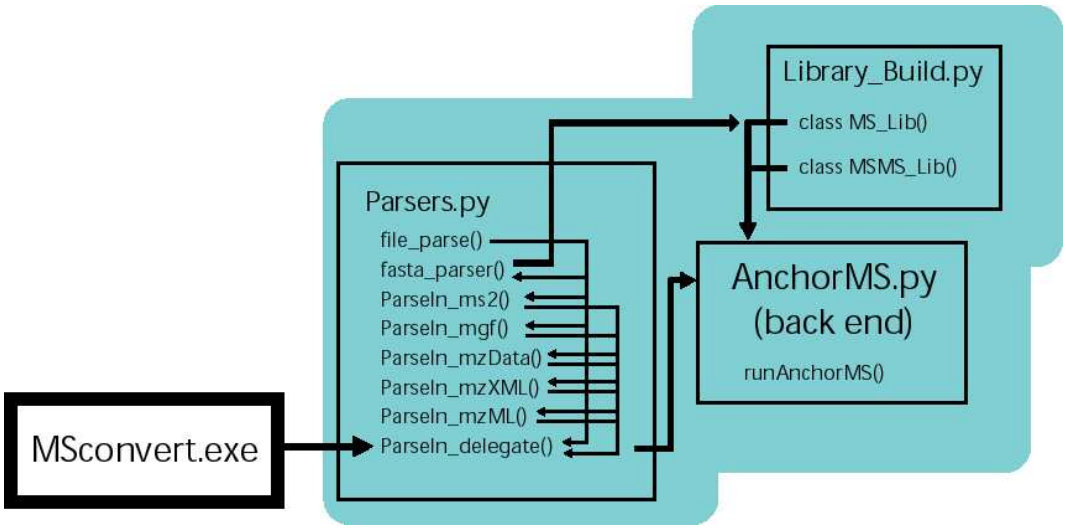


Figure 2.4: A diagram showing the organization of code within the Parsers.py module.

2.3.4 Experimental mass spectrum quality

Sometimes experimentally captured mass spectra are of a low quality, containing high levels of noise or numerous contaminant peaks. Currently, AnchorMS does not exclude noise peaks from low quality spectra, nor does it in any way, "clean up" the mass spectrum prior to analysis. Therefore, if experimental spectra are of a low quality, it is recommended that the low quality spectra first be processed to remove noise, contaminant peaks and possibly low abundance isotopes prior to AnchorMS analysis, to achieve optimum results. Many instrument platforms automatically remove noise peaks, generating high quality experimental spectra. Nonetheless, third party software packages, such as OpenChrom (Wenig and Odermatt, 2010), WaveletQuant (Mo *et al.*, 2010) or P-BOSS (Morohashi *et al.*, 2007), can also be employed to remove noise peaks from the experimental spectra ahead of an AnchorMS analysis.

2.4. Library construction

2.4.1 Overview of library size

In order to identify possible cross-linked di-peptides in the MS^1 spectrum, a library of all possible di-peptides must be constructed, using the sequences of all the proteins that form part of the complex. These protein sequences are digested *in silico* (Digestion.py), and from the resultant pool of all possible, unique peptide sequences, those with residues that are reactive to the selected cross-linking reagent, are paired (Crosslinking.py). For each cross-linked peptide pair, the possible number and type of post-translational modifications (PTMs) are determined (Modifications.py). All permutations of all possible PTMs (as selected by the user) are calculated for each precursor (Permutations.py). The mass of each peptide sequence is calculated from known amino acid residue masses, and the mass of each PTM and cross-link is calculated based on their chemical formulae and empirically established atomic masses. These masses are used to calculate the mass of each di-peptide precursor.

The MS^2 spectrum is analysed to confirm the identity of each putative di-peptide precursor detected in the MS^1 spectrum, and to determine which residues are cross-linked. In order to evaluate whether the observed MS^2 spectrum is consistent with the putative di-peptide precursor, a library of possible fragment peaks is constructed for comparison. For each di-peptide all possible positions for the cross-linked residue within each peptide sequence are calculated (Crosslinking.py). Each such di-peptide

precursor, with its unique set of cross-linked residues, is CID fragmented *in silico*, generating distinctive theoretical spectra of expected MS^2 fragments (Fragmentation.py).

2.4.2 Restriction of library size

The combinatorial space of all possible species calculated in this way can be immense, where there are many post-digest peptides or where many modifications are considered. For example, human histone H3 is rich in arginine and lysine residues, which are sensitive to cleavage by the protease trypsin and prone to numerous PTMs, including mono-, di- and tri-methylation of lysine, mono- and symmetric or asymmetric di-methylation of arginine, acetylation of lysine, phosphorylation of serine, threonine or tyrosine and ubiquitination of lysine. However, in practice, only a fraction of expected species are typically observed. Certain subsets of precursor species may occur at a far lower incidence than expected because they are created by rare events. Some fragments do not retain charge well, and will therefore go unobserved in MS. Rare species can be excluded to reduce library size and render library construction computationally tractable.

AnchorMS allows for the exclusion of a number of low-incidence species during library construction: a protease will occasionally fail to cleave a peptide at a susceptible residue, resulting in a larger peptide. Peptides bearing a greater number of PTMs require a greater number of modification events, and are thus theoretically less likely to be observed. Particular categories of PTM, such as phosphorylation of tyrosine, are also seldom observed. These parameters can be set by the user to adjust how inclusive of low-incidence species an AnchorMS analysis is. Some packages, such as MSX3D, consider the rare formation of 2^{nd} and higher order cross-linking, where three or more peptides are cross-linked. AnchorMS excludes these precursors from library construction, and only considers 1^{st} order cross-linking.

2.4.3 Implementation and data structures

In AnchorMS the module Library_Build.py is used to construct the theoretical libraries. The Library_Build.py module defines two python objects, *MS_Lib* and *MSMS_Lib,* which serve as data structures to house the constructed precursor and fragment libraries, respectively.

Each precursor or fragment component of the precursor and fragment libraries is

associated with a unique numerical identifier which is generated by the module Identifiers.py. Each time a numerical identifier is assigned, its value is incremented. The ongoing increase of the assigned identifier value ensures that each newly assigned identifier value is unique for a particular AnchorMS analysis. The use of unique identifiers also ensures that even very similar library components can be distinguished. For example, two theoretical precursors may have the same residue sequence and cross-linked residues, but differ in their post-translations modifications. If AnchorMS identified theoretical di-peptides by their sequences, then AnchorMS would not be able to distinguish between these two di-peptides.

2.5 Precursor Library Construction (MS^1)

2.5.1 Digestion

During sample preparation for an MS3D experiment, the cross-linked protein complex is digested with a protease, yielding a mixture of peptides. Some of these peptides are cross-linked di-peptides. AnchorMS simulates protease digestion *in silico* using the module Digestion.py, based on which residues in the sequences are susceptible to cleavage by the selected protease. From the uncross-linked protein sequences, all possible post-digestion peptides are calculated. Within the construction of the MS^1 precursor library, cross-linking is performed after digestion using the post-digest peptides.

2.5.1.1 Protease cleavage model

The Digestion.py module attempts to replicate the action of proteases on protein sequences. Protease enzymes cleave a subset of inter-residue peptide bonds within a protein. The bonds that are cleaved in a protein is sequence sensitive and depends on where the enzyme binds. Protease enzymes typically bind to one or more specific recognition sequences. Recognition sequences generally consist of a single residue, but can be longer. For example, trypsin will bind to lysine (K) or arginine (R) residues (Siepen *et al.*, 2007) while Granzyme B will bind to the sequence IEPD (Thornberry *et al.*, 1997).

The exact cleavage site is consistently positioned relative to the recognition sequence, and may vary between different proteases. The vast majority of commonly used proteases, including trypsin, will cleave the peptide bond on the immediate C-terminal side of the recognition sequence. Thermolysin is a notable exception, and cleaves the

peptide bond on the N-terminal side of hydrophobic residues (Ambler and Meadway, 1968; Olsen *et al.*, 2004).

Figure 2.7A below depicts the relationship between the recognition sequence and cleavage site, as implemented in the AnchorMS protease model. In the Digestion.py module of AnchorMS, cleavage sites are defined relative to the peptide bond immediately on the N-terminal side of the residue that is closest to the protein N-terminus within the recognition sequence. This point serves as a "zero-value" reference point. The cleavage site is represented as a cleavage site value which denotes the number of residues between the cleavage site and the "zero-value" reference point. Cleavage sites cleave on the C-terminal side of the reference point are positive and those on the N-terminal side are negative. Therefore, a protease such as thermolysin is said to have a cleavage site of "0" and proteases like trypsin are said to have a cleavage site of "1".

Protease digestion does not always proceed with perfect efficiency. Typically, all susceptible sites in a protein sequence are cleaved. However, occasionally the protease will fail to cleave (miss-cleave) a susceptible peptide bond. Protease miss-cleavage can be promoted by a number of factors, including the presence of proline residues N-terminal to the miss-cleaved peptide bond (Waugh, 2011), fluctuations in pressure (Bonneil *et al.*, 2010) or the presence of residue modifications (including peptide cross-links). In the case of the latter, protease recognition of specific sequences is highly specific and thus disrupted by the changes to a residue resulting from a PTM. The lower the digestion efficiency of the protease under the experimental conditions, the greater the maximum number of missed cleavages one can expect to observe in post-digestion peptides. In practice, protease digestion rarely generates peptides with more than 2 miss-cleavage sites. AnchorMS uses 1 missed cleavage as a default value.

2.5.1.2 Module organisation

Figure 2.5 below summarizes the organization of the Digestion.py module. Within the Digestion.py module, the *digest()* function is called to perform *in silico* digestion on protein sequences. The *digest()* function also makes use of the functions *derive_cut_sites()* and *enzyme_parse()*. The *derive_cut_sites()* function identifies all potential protease cleavage sites within a protein sequence. The *enzyme_parse()* function serves as an internal parser for *digest()*, screening against invalid parameter

values and interpreting parsed information. The Digestion.py module also contains protease information for a number of commonly used proteases with established recognition sequences and cleavage site values (Rawlings and Barrett, 2000; Rawlings and Salvesen, 2012), including trypsin, glu-C and thermolysin. Information for custom proteases defined via the web interface for a particular analysis is written to CustomProteasesFile.py file and imported by the Digestion.py module.

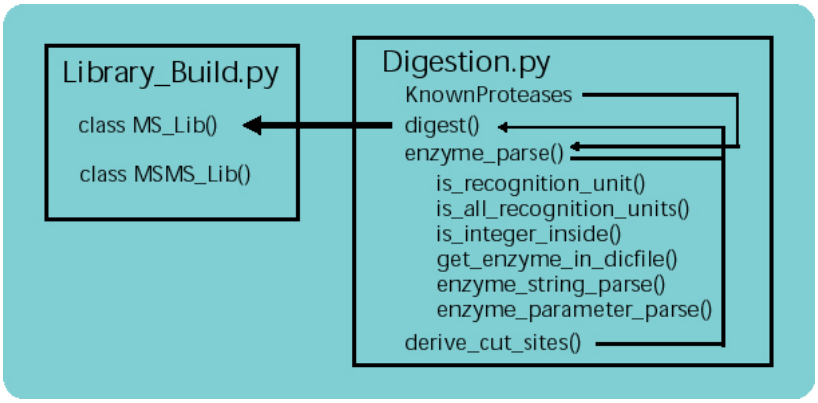


Figure 2.5: Diagram showing the organisation of functions within the Digestion.py module.

2.5.1.3 Parsing and parameters

The *digest()* function accepts three parameters, namely the amino acid sequence, protease information, and the maximum number of miss-cleavages permitted per peptide.

The protease information includes both recognition sequences and cleavage site +protease recognition sequence(s) as strings (see Figure 2.6 below). Each list element carrying a recognition sequence may be followed by the cleavage site value as an integer. Where no integer value follows a recognition sequence string, the cleavage site value defaults to “1” since this is the most common value amongst known proteases.

Similarly, the optional third *digest()* parameter defining the maximum number of missed cleavages defaults to “1” since this represents a typical digestion efficiency for known proteases. By specifying the maximum number of missed cleavages to be considered per peptide, calculations can be constrained to generate only the most likely post-digestion peptides and result in a shorter overall computation time.

At the end of *in silico* digestion a Python list of post-digest peptide sequences is returned.

```
[<recognition sequence 1>, <recognition sequence 2>, .. ]
[<recognition sequence 1>, <cleavage site value 1>, <recognition sequence
2>, <cleavage site value 2>, .. ]
Eg.: [ 'K', 'YH', 0, 'R', 1 ]   --→  Cleaves C-terminal to lysines &
arginines, and N-terminal to tyrosines in 'YH'.
```

Figure 2.6: Example of valid parameter inputs for the *digest()* function in the Digestion.py module.

2.5.1.4 Algorithm and workflow

An *in silico* digestion is initiated by calling the *digest()* function. Using the *derive_cut_sites()* function, the protein sequence is traversed to find each occurrence of any of the protease recognition sequences. Python's string search is employed here and implements a form of the Boyer-Moore-Horspool searching algorithm (Boyer and Moore, 1977; Horspool, 1980; Hume and Sunday, 1991). The position of each recognition sequence is stored as a string index in a temporary Python list, and redundant entries removed. Each of these recognition sequence positions is adjusted according to the cleavage site value, creating a Python list of cleavages sites within the protein sequence.

As shown in Figure 2.7B below, the residues between each cut site are extracted as sub-sequences that represent the peptide sequences that result from a 100% efficient digestion (without missed cleavages). This list of sub-sequences is ordered in the order in which they are found in the protein sequence, from N-terminus to C-terminus.

In order to imitate missed cleavages, sub-sequences which are adjacent in the protein sequence and on either side of a missed cleavage site are concatenated to form the peptide sequence that results from a missed cleavage at that site. The list of sub-sequences is traversed by moving an increasingly larger window across the list, and concatenating the sub-sequences within that window (see Figure 2.7B). The maximum window size will be one greater than the maximum number of missed cleavages. Each window size corresponds to a particular number of missed cleavages, as shown in Figure 2.7B. The sub-sequences within each window are concatenated to form a peptide sequence that is appended as a string to a Python list that already contains the original sub-sequences.

This Python list is the complete set of expected peptide sequences given the protease information and the maximum number of missed cleavages. This Python list is returned by *digest()*.

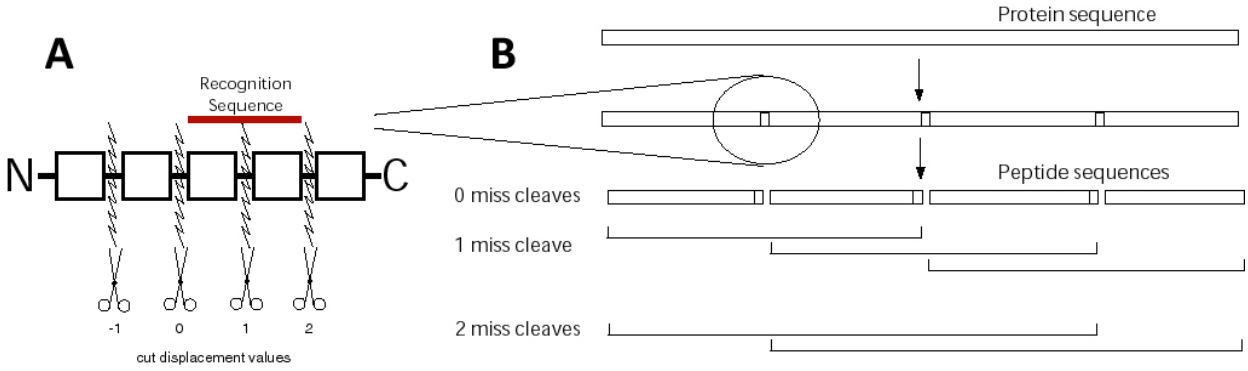


Figure 2.7: The operation of the *digest()* function in the AnchorMS module Digestion.py. (A) The *digest()* function cleaves the protein sequence at a position relative to the recognition sequence of the protease. A cut displacement value is used to represent this and is measured relative to the inter-residue bond on the immediate N-terminal side of the recognition sequence, as shown. (B) The *digest()* function calculates the peptide sequences produced by incomplete digestion of a protein sequence where proteases cleave at specific short sequences.

2.5.2 Cross-linking

During the sample preparation for an MS3D experiment, the purified protein complex is treated with a cross-linking reagent while still in its native conformation. As a result, the post-digestion mixture contains a number of cross-linked di-peptides, each of which can provide valuable structural information about the native conformation of the protein complex.

The Crosslinking.py module simulates the action of a cross-linking reagent *in silico*. Based on the set of post-digest peptides generated by the Digestion.py module, Crosslinking.py calculates the set of all unique, di-peptide precursors possible.

2.5.2.1 Cross-linking structural information

Within the native conformation, two residues can only be bridged with the cross-linking reagent during incubation if two criteria are met. Firstly, each residue must be reactive with the respective functional groups of the cross-linking reagent. Secondly, the distance between the residues in the native conformation of the protein complex must be equal to, or less than, the unfolded length of the molecular spacer arm between the two functional groups of the cross-linking reagent. Any two residues which are found to be cross-linked in the sample must be within this distance. Spacer arms within a cross-linking reagent that differ in length will also impose different distance constraints. In order to obtain more detailed structural information, and test for a number of inter-residue distances, a protein complex can be treated with multiple cross-linking

reagents, each bearing a spacer arm of a different length. For example, EDC will only cross-link contacting residues (zero distance) while BS^3 can cross-link residues up to 11.34 Å apart (Chen *et al.*, 2010; Sinz, 2003). Figure 2.8 below illustrates this principle.

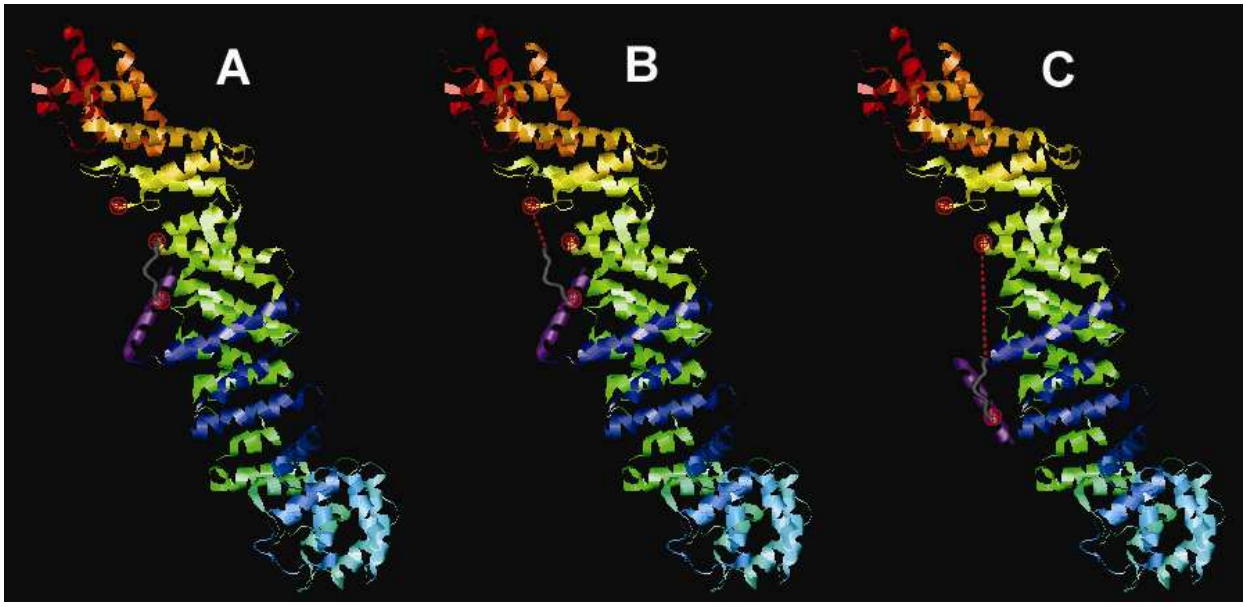


Figure 2.8: The structural significance of cross-linked residues in a di-peptide precursor. In the three scenarios above (panels A – C), red target symbols indicate sites chemically reactive to the cross-linking residue and the spacer arm of the cross-link is depicted in dark gray. Where cross-linking is not possible the space between the target amino acid and the reactive end of the cross-linking reagent is highlighted with a red, dotted line. The diagram above considers scenarios where a theoretical α helix (in purple), appended to the N-terminal of nucleoporin NIC96 (Schrader *et al.*, 2008), assumes one of two orientations (panels A and B, and panel C, respectively). The cross-linking reagent imposes a particular distance constraint on the cross-linked residues in the native conformation based on the unfolded length of the molecular spacer arm between the functional groups. The cross-link seen in Panel A, is not possible if the appended α helix (in purple) is in the alternative orientation (Panel C). The detection of this cross-link excludes structural models where the appended helix is orientated as shown in Panel C.

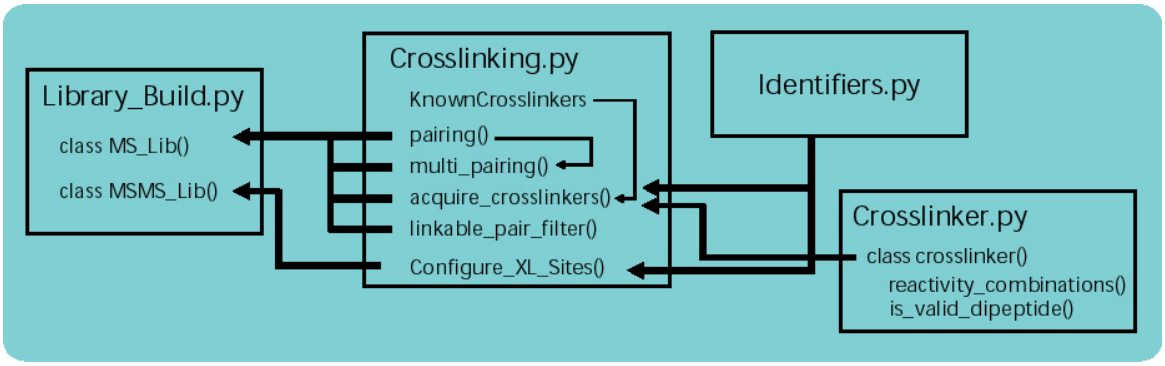


Figure 2.9: Diagram showing the organisation of the Crosslinking.py module.

2.5.2.2 Module organization

Collectively the functions in the Crosslinking.py module generate a series of cross-linked di-peptides based on a peptide list and one or more specified cross-linking reagent. The Crosslinker.py module defines a cross-linker object that is used by Crosslinking.py to store information relating to individual cross-linking reagents. Figure 2.9 outlines the organisation of the Crosslinking.py and Crosslinker.py modules.

Crosslinking.py is used by the Library_Build.py module in the construction of both precursor (MS^1) and fragment (MS^2) libraries.

The first four functions are used for precursor library construction. The *linkable_pair_filter()* function excludes peptide pairs that lack residues reactive the cross-linking reagent, making use of the *residue_partnerships()* method of the cross-linker object. The *pairing()* and *multipairing()* functions generate a set of all unique, cross-linkable peptide pairs possible. The *acquire_crosslinkers()* function serves as an internal parser for cross-linking reagent information and instantiates cross-linker objects. The *Configure_XL_Sites()* function is used only for fragment library construction and calculates the set of all possible pairs of cross-linked residues within each di-peptide precursor.

Crosslinking.py also stores the information for a number of commonly used cross-linking reagents, each of which can be selected by the user through the web interface. Information for custom cross-linking reagents defined by the user through the web interface is stored in the CustomCrosslinkerFile.py, and imported into the Crosslinking.py module.

2.5.2.3 Algorithm and workflow

The *in silico* cross-linking of post-digest peptides is performed in two independent stages. In the first stage, *in silico* “cross-linking” is performed by constructing the MS^1 precursor library, and involves the pairing of peptides that may form di-peptide precursors. In the second stage, *in silico* “cross-linking” is performed by construction of the MS^2 fragment library, and involves defining the cross-linked residues of di-peptide precursors. *In silico* cross-linking for precursor library construction is performed on a set of peptide sequences, parsed in as a Python list of sequence strings. Each of the peptide sequences is paired with every peptide sequence, including itself.

The algorithm can be set to assemble only inter-molecular pairs where each peptide in

the di-peptide originates from a different protein subunit in the protein complex, and exclude intra-molecular pairs where both peptides originate from the same protein. Disabling the pairing of intra-molecular peptides reduces AnchorMS analysis time by restricting the analysis to include only those di-peptides that may provide inter-subunit structural information about the protein complex. Because such structural information is considered to be of greater interest in an MS3D experiment, this is the default setting.

As each peptide pair is assembled, the *linkable_pair_filter()* function determines if the peptides can be cross-linked based on the cross-linking reagent. For homobifunctional cross-linking reagents there must be at least one reactive residue on both peptides for a di-peptide to form. For heterobifunctional cross-linking reagents, at least one residue reactive to one of the functional groups and at least one residue reactive to the other functional group should be present on different peptides for a di-peptide to form. For example, a heterobifunctional cross-linker with non-overlapping specificities cannot cross-link identical residues. Each theoretical peptide pairing is checked according to these criteria for valid residues. Those theoretical peptide pairings without appropriate reactive residues in each peptide are excluded from di-peptide library construction.

2.5.2.4 Data structures and parsing

Information relating to each cross-linking reagent is stored in a cross-linker object, defined in Crosslinker.py. The Crosslinking.py module makes use of the 'cross-linker' class, instantiating an object for each cross-linking reagent selected or defined by the user. Name, formula, spacer length, linked mass, formula of atoms displaced during bonding and amino acid reactivities are included and assigned to the object.

The Crosslinking.py module features an internal parser which checks for invalid input and accepts cross-linker objects as well as a string or list of strings. Each string is treated as a file path for a tab or comma delimited text file containing the parameters for a cross-linking reagent on each line. Amino acid reactivities for each functional group are enclosed in brackets, and separated by commas as in, for example, “('K','R')”. Figure 2.10 below depicts this structure and shows an example of a typical entry defining a cross-linking reagent. The user has the option of including the linked mass directly or including only the atomic formula of the cross-linker and the atomic formula of the atoms in the peptide displaced during the cross-linking reaction. In the case of the latter, the linked cross-linker mass will be calculated from these two

supplied atomic formulae. Reagent names must be preceded by a ">" to distinguish them from comments.

```
<reactivity 1>; [<reactivity 2>]; [<reactivity 3>]; <spacer arm length>;
<unlinked formula> OR <linked mass>; [<displaced atoms formula>];
[<free mass>]; [<released mass>]; [<cleaved mass>]; [<name>]; [<comment>]
"(E,D);(K);0.0;-18.010565;>EDC,concentration=5.0mM"
```

Figure 2.10: The order of parameters relating to the cross-linking reagent and the 'crosslinker' object. Optional parameters are placed in square brackets. Lines 2-4 present examples.

2.5.2.5 Cross-linking within library construction

During the construction of the precursor library for MS^1 analysis, the output is "site-blind". That is, (at this stage in the workflow), the sites of cross-linking within each peptide are not included or defined for each di-peptide precursor. For the theoretical MS^1 spectrum, only the mass of possible precursor molecules are calculated. This precursor mass is unaffected by the sites of cross-linking within the di-peptide precursor. Within the precursor library construction workflow, "sight-blind", *in silico* cross-linking results in a set of peptide sequence pairs, each associated with a particular cross-linker object.

However, during the fragment library construction, for MS^2 analysis, the site of cross-linking (the cross-linked residues) within the di-peptide precursor is considered. In each "site-blind" theoretical di-peptide precursor in the MS^1 precursor library, there may be multiple residues in each di-peptide which can be cross-linked to the other peptide. Each inter-peptide pair of residues that can be cross-linked constitutes a distinct theoretical precursor within the theoretical MS^2 fragment library. Unlike the precursor mass, the MS^2 spectrum produced by a di-peptide precursor depends on which residues (and their position) in the di-peptide are cross-linked. For this reason, *in silico* cross-linking during fragment library construction for MS^2 analysis may be termed "site-sensitive", rather than "site-blind". In order to model all possible MS^2 spectra that may be observed for each MS^1 precursor peak, all possible combinations of cross-linked residues must be considered. Within the fragment library construction workflow, "site-sensitive" *in silico* cross-linking yields a set of all possible cross-linking sites within each MS^1 di-peptide precursor. For each of these di-peptide precursors, defined by both the sequences of the peptides as well as its cross-linked residues, the Fragmentation.py module will generate a theoretical MS^2 spectrum.

2.5.3 Modifications

Cellular enzymes modify residues within the protein molecule *in vivo*, typically by the addition of a chemical group, such as phosphate ($-PO_3$). The chemical modification of protein residues can also be induced during sample preparation. For example, thiol bearing cysteine residues may be alkylated (CH_2CH_3) by treatment with iodoacetamide to prevent the formation of inter-cysteine di-sulfide bridges. These modifications may survive the cross-linking and digestion processes, and add to the mass of the resultant di-peptide precursor molecules.

In order to define all possible precursors, all possible modification states must be defined as well. In AnchorMS, the Modifications.py module calculates the possible permutations of chemically induced and post-translational modifications for a given di-peptide precursor.

2.5.3.1 Module organisation

The Modifications.py module is used by the Library_Build.py module during the construction of the theoretical precursor library to add possible induced and post-translational modifications to each di-peptide precursor. Figure 2.11 summarizes the organisation of the Modifications.py module. Library_Build.py calls the *getModState()* and *sum_PTM_displace()* functions. The *getModState()* function in turn makes use of the *get_possible_mods()* function as well as the *get_permutations()* function from the Permutation.py module.

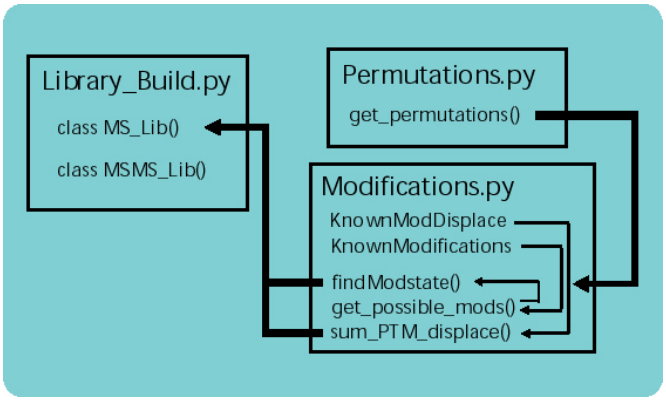


Figure 2.11: The organization of functions and data structures in the Modifications.py module. Arrows from a function point to calling functions.

Known, commonly observed modifications for each residue type are stored in the Modifications.py module as a Python dictionary, 'KnownModifications'. In addition, modifications defined by the user through the web interface are stored in

CustomModificationsFile.py for each session, and imported into Modifications.py as the Python dictionary ‘CustomModifications’.

2.5.3.2 Combinatorial space and library size restriction

2.5.3.2.1 Fixed and variable modifications

For the purposes of enumerating the possible di-peptides and constructing a library of theoretical precursors, there are two types of modifications, namely fixed and variable modifications. If a protein is chemically modified with 100% efficiency, whether *in vivo* or *in vitro*, then it is termed a fixed modification. Fixed modifications do not expand the combinatorial space of possible di-peptides because there is only one modification state to be considered for each residue. If a modification is added to a protein with less than 100% efficiency and affects only a subset of susceptible residue types, then any given residue of that type may or may not carry the modification, and both possibilities must be considered. This is termed a variable modification. Typically, chemical modifications induced during sample preparations are fixed while PTMs generated *in vivo* tend to be variable. Chemically induced modifications that target residues susceptible to PTMs that are likely to be present in the sample should not be considered as fixed modifications, because residues already carrying the PTM will not incorporate the induced modification.

2.5.3.2.2 Combinatorial space expansion

Computing the permutations of variable modification states that are possible for each di-peptide precursor is the most combinatorially intensive component of the precursor library construction process. As represented in Figure 2.13 and approximated in Equation 2.1, the number of modification state permutations in a di-peptide precursor increases exponentially (~M^N) in terms of the number of modification types and the number of modifiable sites, respectively. The Unimod database of post-translational modifications (PTMs) features a large number of possible modifications for a number of amino acid residues, including many PTMs that are seldom observed. Therefore, an exhaustive analysis considering all PTMs for each residue would have a very large M value. This would result in an explosion in the size of the theoretical precursor library, and extend computation time even though many of the PTMs considered are unlikely to be observed.

Equation 2.1: $$Modification\ permutations = \left(Modification\ types \right)^{modification\ sites} = M^N$$

2.5.3.2.3 Combinatorial space constraint

The combinatorial space can be constrained in two ways. Firstly, since many classes of proteins are known to bear specific modifications, it may be realistic to limit the modifications considered to those specific types of modifications. In AnchorMS the user selects those modifications which they would like to include in the analysis and can define custom modifications as well. Secondly, the protein may be known to be prone to a relatively low density of PTMs, and so a maximum number of modifications considered per peptide can also be set. In AnchorMS the default value for this is 2 modifications per di-peptide. The user can set the maximum number of modifications allowed per di-peptide to an arbitrarily large value, such as 100, to include the highest modification density chemically possible, but this may cause an excessive increase in computation time.

2.5.3.3 Algorithm and data structures

2.5.3.3.1 Possible modifications

The function *get_possible_mods()* calculates the possible modifications for each residue in a theoretical di-peptide precursor. The sequence of each di-peptide is traversed and for each residue the possible modifications to be considered are determined and collated into a Python list. Each element in the list represents a possible modification, stored as a string containing the chemical formula of the modification. However, the first element of the list is not a string but a Python null value ("None") indicating the absence of any modification on that residue. The modification lists for each residue is placed inside another Python list according to the order of the residues in the di-peptide sequence. This data structure is summarized in Figure 2.12 below.

```
[ [<residue 1 modification list>], [<residue 2 modification list>], .. ]
[ [None,<mod formula 1>,<mod formula 2>,..], [None,<mod formula 1>,..], ..]
Eg.: [ [None,'PO3H2']], [None], [None,'PO3H2','SO3'], [None],
[None,'CH3','C2H5'] ]
```

Figure 2.12: Representations of the modifications as implemented in the Modifications.py module.

2.5.3.3.2 Permutations of modification states

Once the modifications possible for each residue in the di-peptide sequence have been calculated, the permutations of possible modifications for the di-peptide sequence as a whole needs to be defined in accordance with the supplied combinatorial constraints.

The *get_permutations()* function in the Permutations.py module (see section 2.5.4 below) is used by the *getModStates()* function to calculate these permutations, based on the valid modifications for each residue type and constrained by the maximum allowed number of modifications per precursor molecule.

The *get_permutations()* function accepts a list of integers where each integer is the maximum value for the element at that position in the list. In order to conform to the parameter format required for *get_permutations()*, the list of lists representing possible modifications at each site is converted into a list of integers of the same length as the sequence. In this list, each list element represents a residue in the sequence and each integer denotes the number of modifications possible at that position in the sequence.

This list is parsed into the function *get_permutations()* which generates all possible modification states as a list of lists, each in a similar numeric format. The numeric format is converted back into a list of chemical formula strings by interpreting each integer in the numeric format as a list index for the list of modification strings (Figure 2.12). The *getModStates()* function returns this list for use in precursor library construction by the Library_Build.py module.

2.5.3.3.3 Molecular weight calculations

All of the residue modifications add a chemical group to the modified residue during formation. This reaction may displace atoms from the original chemical structure of the modified residue. For example the addition of a sulfate group (SO_3) displaces a hydrogen atom (H). The displaced atoms for each of the selectable, known modifications are stored in a Python dictionary as a chemical formula string in the Modifications.py module.

In the Library_Build.py module the mass of each di-peptide precursor is calculated as part of precursor library construction. The mass of the unmodified di-peptide is determined using the *protMW()* function in the Bits.py module based on the cross-linking reagent and the amino acid sequences of each strand. The mass of the modified di-peptide precursor is obtained by adding and subtracting, from the unmodified di-peptide mass, the mass of the added and displaced chemical groups, respectively. Each added and displaced mass is calculated from the chemical formula of each using the *atomMW()* function in Modifications.py.

2.5.3.3.4 Modifications within library construction

For the purposes of calculating modification states for MS^1 library construction, each di-peptide is considered to be cross-linked at the first residue in its sequence which is reactive to the cross-linking reagent (as a preliminary assumption). Any necessary corrections will be effected during MS^2 library construction (see section 2.7.1). The determination of the possible modification states of each theoretical di-peptide precursor concludes the construction of the precursor library. This library will be used for MS^1 analysis and matched against the observed MS^1 spectrum to identify putative di-peptide precursors therein.

2.5.4 Permutations

Several tasks within AnchorMS require the enumeration of all permutations in a combinatorial space. For example, the determination of all possible precursor modification states for each di-peptide precursor involves a permutation calculation. In AnchorMS, this generic calculation is carried out by the *get_permutations()* function in the Permutations.py module.

2.5.4.1 Numerical representation of permutations

During library construction, each permutation calculation applies different information. For *get_permutations()* to calculate the possible permutations in each case, permutations in the combinatorial space must be represented in a standard form. Figure 2.13 illustrates how permutations are represented in AnchorMS for the purposes of calculating the possible permutations.

Permutations are calculated for systems that contain a series of unit entities which exist in a particular order and can each assume one of multiple states. For example, in a peptide precursor, the order of the constituent residues (ordered unit entity) is significant and each residue may carry one of many post-translational modifications (its modification state). In Permutation.py, each permutation is represented as Python list, and the ordered unit entity is represented by a particular element within that Python list. The order of the elements in the Python list reflects the order of the ordered unit entities within the overall system. Each state which a given ordered unit entity can assume is represented as an integer value, stored at the element position of its ordered unit entity within the Python list (see Figure 2.13A). Each state or permutation of the overall system, is represented as such a list of integers. When *get_permutations()* has calculated all possible permutations of a system, it returns a list

of such integer lists (see Figure 2.13B). Generally speaking, permutations in a combinatorial space are sometimes described using a tree structure diagram. Figure 2.13C features such a tree structure diagram and illustrates that the "list-of-integers" representation, depicted in Figure 2.13B, is equivalent to a tree structure representation.

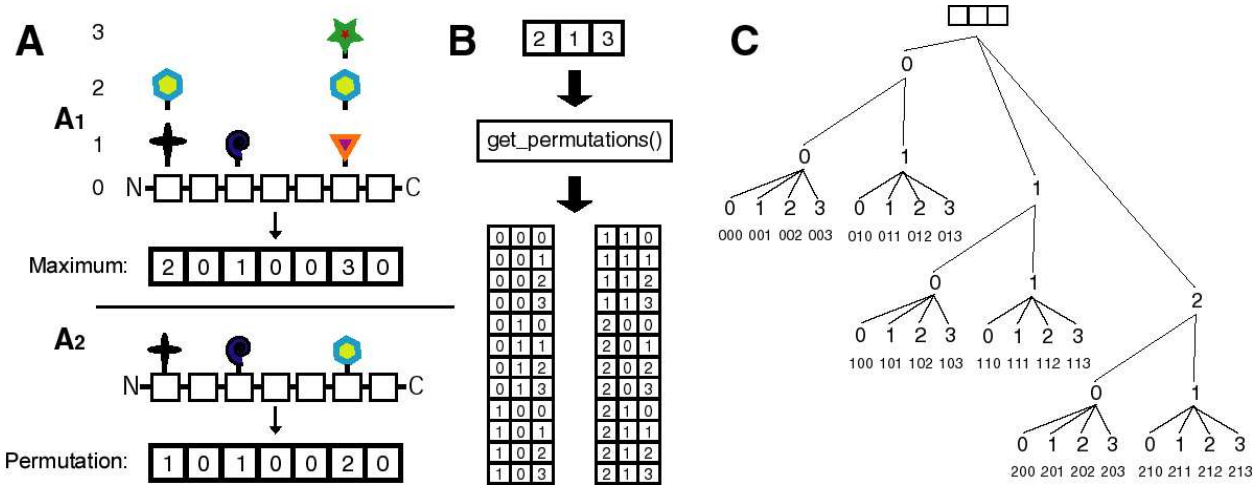


Figure 2.13: A diagram depicting the purpose of the *get_permutation()* function. (A) The example of peptide modification states illustrates how a combinatorial space is represented numerically for the purposes of calculation by *get_permutations()*. (B) From a set of maximum values *get_permutations()* generates all possible sets of values. (C) A branching tree representation of the sets of values generated by *get_permutations()*.

2.5.4.2 Parameters of get_permutation()

2.5.4.2.1 The number of states for each ordered unit

In order to calculate all possible permutations, the number of states in which each ordered unit entity can exist needs to be defined. The *get_permutations()* function accepts a list of integers as the first parameter, which is similar to the list-of-integers representation of permutations, but contains slightly different information. Each list position in the parsed list corresponds to an ordered unit entity, and the integer values define the maximum values at that list position (i.e., for that particular ordered unit entity) in a "list-of-integers" representation (see Figure 2.13A, above). Therefore, the element value at each list position is one less than the number of states in which that ordered unit entity can exist (because element values begin at zero).

2.5.4.2.2 A maximum sum of state values for the overall system

For some systems, each state that an ordered unit entity can assume is defined by quantitative categories. For example, the number of simple sequence repeats (SSRs) at a particular locus of a chromosome defines a state of that locus in terms of a quantitative category (An example unrelated to AnchorMS). The *get_permutations()* function allows for the exclusion of overall system states where the sum of state values (for all its ordered unit entities contained therein) exceeds a particular value.

As an optional second parameter, *get_permutations()* accepts an integer which defines the maximum value for the sum of all element integer values in a list that represents a permutation.

2.5.4.3 Algorithm implemented for calculating permutations

The possible permutations for a system are calculated by *get_permutations()* through a simple algorithm (Figures 2.14 and Figure 2.15). The implemented algorithm was found to be quicker than a recursive algorithm, but may not achieve the maximum speed theoretically possible for the calculations of permutations.

The *get_permutations()* function generates successive permutation-defining integer lists by incrementing the appropriate single list element values, while avoiding invalid integer lists. After each element value increment, the resultant permutation-defining integer list is stored in a temporary Python list.

The algorithm begins by incrementing the right-most element (maximum list index). The right-most element is repeatedly incremented until its maximum value is reached. For any list element, when a list element reaches its maximum value, the function will move to the next element, on the its left, and check if the value at this list element is at its maximum. If the checked list element is at its maximum value, then the function will continue moving to the left – checking each list element value – until a list element with a value less than its maximum is found. If the checked list element is less than its maximum, then a) it is incremented by one, b) the values of all the list elements to its right are returned to zero, and c) the function proceeds, again, to repeatedly increment the right-most list element by one. If, in moving to the left within the list, the left-most list element is reached and found to be at its maximum, then the function ends and returns the permutations it has recorded.

A list of lists is returned by *get_permutations()*, wherein each of the inner lists

represents a permutation. Cumulatively, this list of lists represents all permutations within the constraints given to *get_permutations()*.

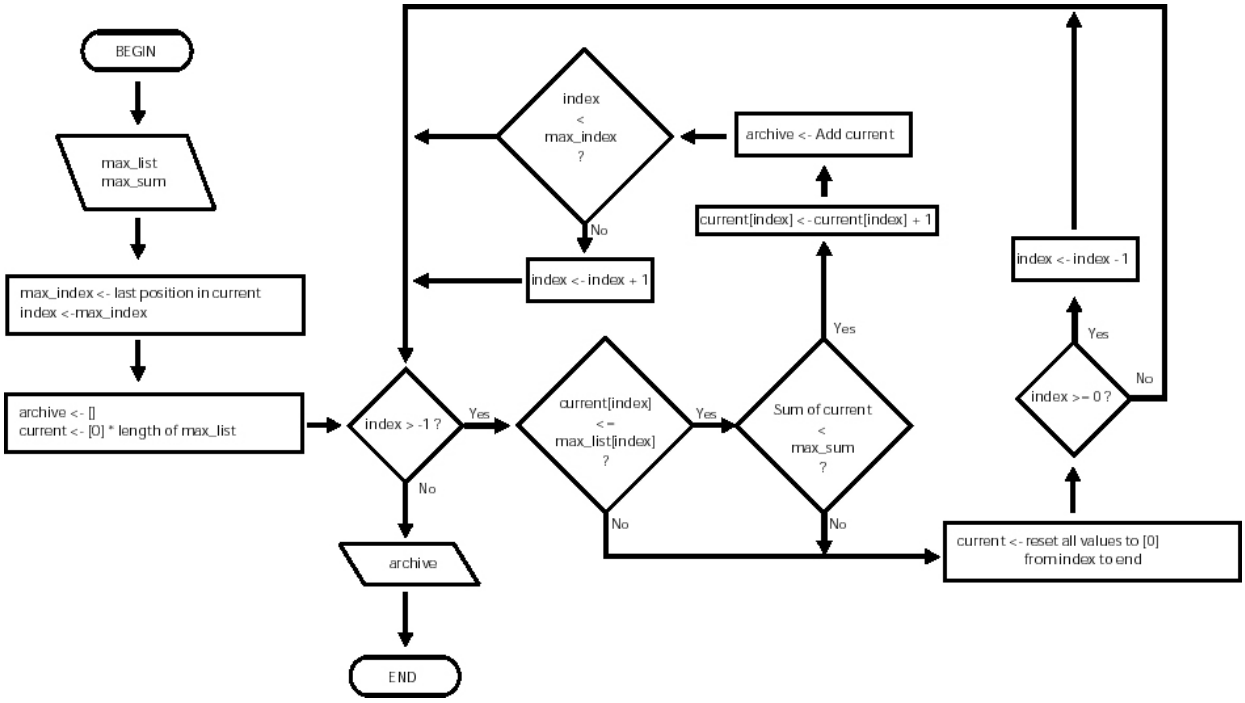


Figure 2.14: A flow diagram of the algorithm implemented in the *get_permutations()* function from the Permutation.py module.

```
1 def get_permutations(max_values, zero=False, max_sum=None):
2     #-- Instantiations
3     archive = []
4     current_list = [0] * len(max_values)   # list of current values
5     mi = len(max_values) - 1        # maximum list index
6                 current = [0]*len(max_quantities)
7     if max_sum == None:
8         max_sum = sum(max_values) + 1
9     #-- Calculations
10    if zero == True:
11        archive.append(current_list[:])
12    while ci > -1:
13        if (current_list[ci] < max_values[ci]) and (sum(current_list)
             < max_sum):
14            current_list[ci] = current_list[ci] + 1
15            archive.append(current_list[:])
16            if ci < mi:
17                ci = ci + 1
18        else:
19            current_list[ci:] = [0] * (mi - ci + 1)
20            if ci >= 0:
21                ci = ci - 1
22    return archive
```

Figure 2.15: Python code defining the function *get_permutations()* from the Permutation.py module without in-code comments.

2.5.5 Generating the predicted mass spectrum: mass and charge

In the sections above we have described the construction of a library of theoretical or predicted precursors. Once a library of theoretical precursors has been constructed, it is used to generate a predicted MS^1 precursor mass spectrum (for comparison with the observed MS^1 mass spectrum). A mass spectrometer generates a mass spectrum by very accurately measuring the mass-to-charge ratio of each analyte ion that passes through the instrument to its detector. Each peak in the mass spectrum represents a detected analyte ion, which denotes its mass-to-charge ratio (*m/z*). Therefore, to correctly populate the theoretical mass spectrum, first the mass and then the *m/z* peak value must be calculated for each predicted precursor ion.

2.5.5.1 Module organization and function

In AnchorMS, the module Mass_Charge.py performs calculations related to mass and charge, containing the necessary functions and information. The mass of chemical formulae and amino acid sequences are calculated by the functions *atomMW()* and *protMW(),* respectively. The *peak()* function calculates *m/z* peak values. The function *MaxChargeDetectable()* determines the theoretical maximum charge state detectable for a given instrument. The predicted precursor mass spectrum calculated is stored within the precursor library object, 'MS_Lib', which is defined in the AnchorMS module Library_Build.py.

2.5.5.2 Calculating molecular masses: atomMW()

The *atomMW()* function calculates the molecular weight of a chemical formula based on empirically determined average and mono-isotopic relative atomic masses of each element (Wieser and Coplen, 2011), recorded to the sixth decimal place. Chemical formulae are accepted as strings and must contain element symbols, with each symbol followed by its numerical formula subscript. The absence of a numerical subscript is taken as 1. At present *atomMW()* only supports subscripts up to 99. The use of average or mono-isotopic masses can be specified. The latter is set as the default because, in the context of mass spectrometry, average masses are only suitable for low resolution instruments.

2.5.5.3 Calculating peptide masses: protMW()

The *protMW()* function calculates the molecular weight of an amino acid sequence using known, established mass values for each bonded amino acid and the *atomMW()* function. Bonded amino acid masses are calculated from known amino acid chemical formulae (Mathews *et al.*, 2000). The *protMW()* function considers the bonded form of each amino acid and adds a hydrogen and hydroxide group to the N-terminal and C-terminal end, respectively. As with *atomMW()*, *protMW()* can be directed to use either average or mono-isotopic masses, with the latter set as the default. The *protMW()* function can accept a dictionary representing the fixed amino acid modifications to be considered.

2.5.5.4 Calculating m/z peak values: peak()

The *peak()* function calculates the *m/z* peak value of an analyte peptide ion as per Equation 2.2 below. Typically the ionization of a peptide within a mass spectrometer

occurs through its acquisition of a hydrogen ion or proton (Wysocki *et al.*, 2001). Therefore, the mass of a proton (M_{proton} in Equation 2.2) must be added to the ion mass for every charge added to an analyte peptide species.

Equation 2.2:
$$Peak\ \ value = \frac{Mass + \left[Charge \times\ M_{\ proton}\right]}{Charge}$$

2.5.5.5 Calculating a default maximum charge: MaxChargeDetectable()

Raw mass spectra are often digitally pre-processed prior to analysis to simplify and "clean up" the data. For example, charge deconvoluting software extracts a list of masses of molecular species present from a spectrum of *m/z* peak values (Dobo and Kaltashov, 2001; Jaitly *et al.*, 2009; Kallos *et al.*, 2005; Reinhold and Reinhold, 1992; Zhang and Marshall, 1998). Raw mass spectra typically contain multiple peaks for each analyte ion detected, each corresponding to a different isotope. For singly charged species, a difference of 1 Da separates the peaks of each isotope of a molecular ion. As the charge state of the molecular ion increases this inter-isotope distance decreases. The inter-isotope peak distance is 1/charge. Charge deconvolution software typically makes use of these distances in an isotope peak cluster to estimate the charge state of observed peaks. If the inter-isotope peak distance for an ion is greater than the instrument mass accuracy, then the isotope peaks can be distinguished from each other and the inter-isotope distance can be reliably determined. The charge state is then deemed "detectable". The *MaxChargeDetectable()* function calculates the maximum charge detectable for a given mass accuracy in this way. In AnchorMS, this value is used as a default, maximum number of charges to be considered during the construction of the theoretical precursor library.

2.6. Identification of putative di-peptides in the MS^1 spectrum

2.6.1 Peak matching

In order to analyse the observed mass spectrum, it is compared to the theoretical mass spectrum. The analysis of an observed mass spectrum attempts to determine the identity of each observed peak. An observed peak is identified on the basis of its similarity to a predicted peak in the theoretical spectrum. An important exercise in this process is to decide whether an observed peak and a predicted peak are sufficiently similar to be considered a match.

2.6.1.1 Determining if two compared peaks match

An observed and a predicted peak are considered to be a putative match if their peak *m/z* values, or deduced masses, differ by less than a defined value. This value is referred to as the tolerance. The stringency of peak matching depends on the tolerance value applied. The larger the tolerance value the less similar two peaks need to be to be deemed a match, and so the more likely it is for a false positive match two occur. If matching is too stringent, false negative peak matches may occur.

Tolerance is needed, because an experimentally observed or apparent peak value may differ, to a certain extent, from the predicted or true peak value. Though the capabilities of mass spectrometry (MS) instruments are continually improving, there is a limit to the measurement accuracy of a given MS instrument. Due to instrument noise, the observed value for any analyte ion can vary stochastically. Observed peak values can only be expected to be within the instrument mass accuracy of the true value. With this in mind, the most intuitive value to use as a peak-matching tolerance is the mass accuracy of the instrument. In AnchorMS the tolerance value applied during peak matching is defined by the user, taking the instrumentation into account that was used to generate the MS data.

2.6.1.2 Restricting the number of peak comparisons

In the analysis of a whole mass spectrum, a number of observed peaks must be compared to predicted peaks. If an observed peak were to be compared to all the predicted peaks in the theoretical spectrum in a brute force fashion, the vast majority of compared, predicted peaks would not match, and the computation time would be greatly and unnecessarily increased. In order to minimize computation time, it is helpful to minimize the number of peak comparisons performed and exclude comparisons which cannot result in a match. Typically, the peak values in a mass spectrum are arranged in ascending order.

In AnchorMS, this is exploited to avoid performing peak comparisons that cannot result in a match. Each peak in the observed spectrum is compared to a selected set of adjacent peaks in the predicted spectrum. This set begins with the first predicted peak being matched to the observed peak immediately prior to the observed peak currently being compared. Predicted peaks before this one were too far from the immediately-prior observed peak to match, and even further from the current observed peak. Naturally, the set of predicted peaks compared to the first observed peak begins with

the first peak in the predicted spectrum. During spectrum matching, any predicted peak greater than a peak that has previously failed to match a given observed peak, will not be compared to that observed peak, since it is clearly even further from said observed peak.

2.6.1.3 Module organisation

In AnchorMS peak and spectrum matching is performed by the Compare_Spectra.py module. The organization of the Fragmentation.py module is summarized in Figure 2.16 below. The *match_peaks()* function houses the core functionality within the Compare_Spectra.py module. The *match_peaks()* function compares two mass spectra using a moving window as described above. Within *match_peaks()* a number of trivial functions, such as *addMatch()*, *delMatch()*, *getFragValue(),* access and modify stored matching information.

In time-of-flight (TOF) mass analyzers, the instrument accuracy is dependent on the size of the peak detected, and generally measured in parts per million (ppm) or parts per billion (ppb). In AnchorMS the user can specify the tolerance value in ppm or ppb or in Daltons. Where the tolerance is in terms of ppm or ppb, *get_absolute_tol()* calculates the equivalent tolerance value in Daltons for each peak comparison.

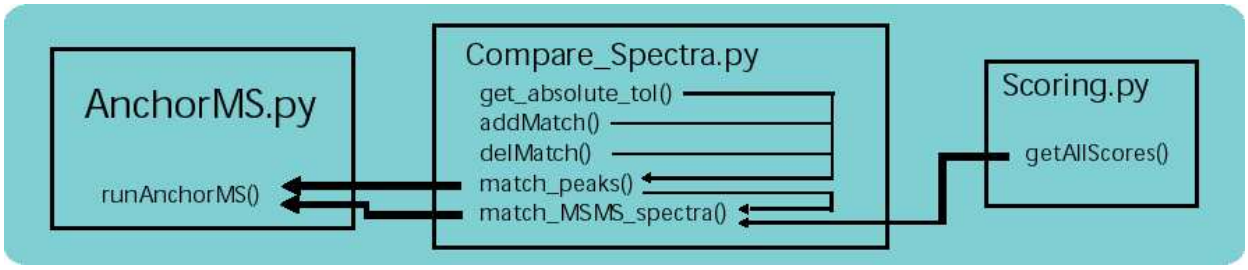


Figure 2.16: A diagram summarizing the organization of code within the Compare_Spectra.py module.

2.6.1.4 Multiple MS^1 candidates and their role in MS^2 analysis

Within an MS^1 spectrum it is possible for two peptides to have very similar peak values due to a similarity in amino acid composition, or by pure chance. As a result, both peak values may be similar to the same predicted peak in the theoretical spectrum. Since both value are within the tolerance value from the predicted peak, both must be considered as a *possible* true match to the predicted peak, even though one is no doubt a false match. In such cases, MS^1 analysis cannot further discriminate between the two alternative peak assignments. MS^2 analysis is required to determine which is likely the true assignment. Even two MS^1 precursors with exactly the same amino acid

composition will generate different fragment spectra if their sequences are different.

Each MS^1 peak that is putatively matched to a predicted di-peptide peak in the theoretical MS^1 spectrum is included in the MS^2 analysis. In AnchorMS, MS^1 analysis serves as a filter to identify *possible* di-peptide species present in the sample. Only those theoretical di-peptides will be included in the constructed, theoretical fragment library. MS^2 analysis confirms matches detected during MS^1 analysis, and determines the site of cross-linking in di-peptide precursors.

2.7. Fragment Library Construction (MS^2)

2.7.1 Cross-link sites and cross-linked residues

During MS^1 precursor library construction, preliminary *in silico* cross-linking was performed in a “site-blind” fashion, where the possible positions of the cross-linked residue in each di-peptide strand were disregarded. The *in silico* “cross-linking” performed for the construction of the MS^2 fragment library is performed by the function *Configure_XL_Sites()* within the Crosslinking.py module, and calculates the set of all possible pairs of cross-linked residues within each di-peptide precursor.

The *Configure_XL_Sites()* function performs *in silico* cross-linking for fragment library construction on a set of peptide sequence pairs which is parsed as a Python list of tuples, each containing two sequence strings. When each 'crosslinker' object is instantiated, all possible combinations of residue pairs that can be cross-linked is calculated based on the residues reactive with each functional group of the selected cross-linking reagent, and stored in the object attribute 'residue_partnerships*'.* The peptide sequences are searched for residue pairs, one in each peptide, that match one of these combinations.

During MS^1 library construction, the calculation of the precursor modification state assumes that the first cross-linkable residue in each peptide strand is cross-linked. In each cross-linked di-peptide calculated by *Configure_XL_Sites()*, if the cross-link involves a residue other than the first cross-linkable residue in the strand, then the modification state of that residue is reassigned to the first cross-linkable residue, as shown in Figure 2.17 below.

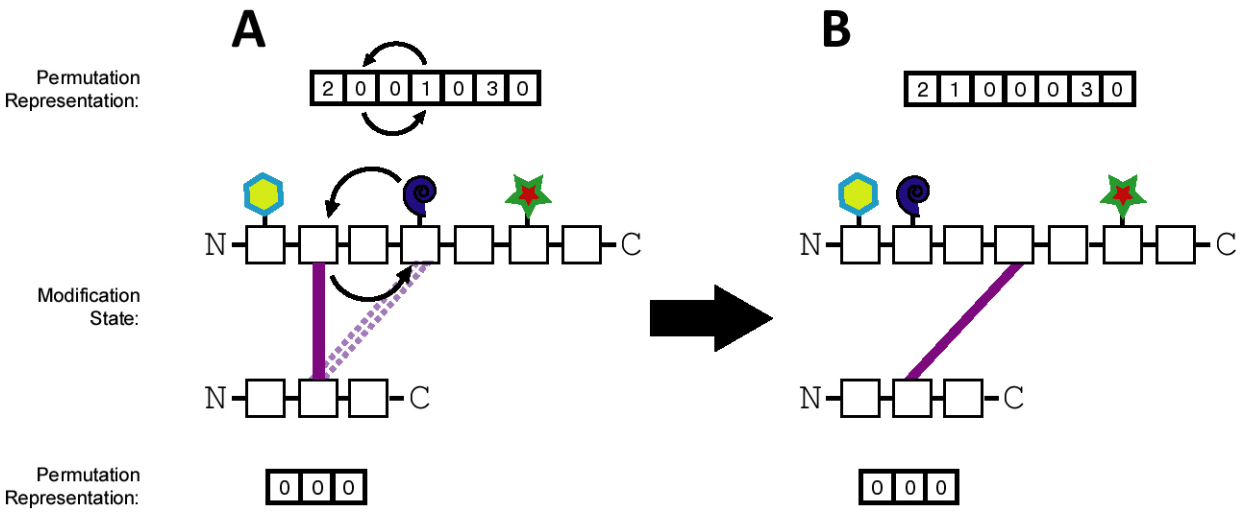


Figure 2.17: A diagram showing how the modification state of cross-linked di-peptide precursors in the MS^2 library is updated to accommodate alternative cross-linking sites during MS^2 library construction.

2.7.2 Fragmentation

Putative di-peptide precursors detected in the MS^1 scan, are passed into a chamber within the mass spectrometer containing an inert gas, such as nitrogen. The collisions between the precursor ion and the gas molecules induces the disruption of inter-residue bonds within the di-peptide precursor, and results in an MS^2 fragment spectrum. The MS^2 spectrum is often specific to its precursor, and is used to identify the precursor, as well as the cross-linked residues in di-peptide precursors, with confidence.

The Fragmentation.py module implements *in silico* collision-induced dissociation (CID) of peptides and di-peptides according to established low-energy fragmentation pathways, as detailed below (Barlow and O'Hair, 2008; Bythell *et al.*, 2010; Paizs and Suhai, 2005). The mass of each peptide in the fragmentations series are calculated from the peptide or di-peptide sequences, modification state and cross-linker mass.

2.7.2.1 Module organization

The Library_Build.py module uses the Fragmentation.py module in the construction of the theoretical fragment library for MS^2 analysis. The *CID_prot_frag()* and *CID_XLprot_frag()* functions in the Fragmentation.py module perform *in silico* fragmentation of single peptides and di-peptides, respectively. *CID_XLprot_frag()* calls *CID_prot_frag()* for each peptide and then adjusts the appropriate fragment mass to account for modifications as well as for the other peptide attached via the cross-link.

Where precursor co-fragmentation is considered, the *get_overlap_spectrum()* and *compile_precursor_overlaps()* functions overlays the theoretical MS^2 spectra of putatively co-fragmented precursors. The *CID_MatchedPrecursors_frag()* and *add_FragPeakDicEntry()* functions assists Library_Build.py in constructing the theoretical fragment library for all detected di-peptide precursors.

2.7.2.2 Fragmentation model and algorithm

2.7.2.2.1 Fragment types

Low energy CID fragmentation occurs in one of three ways, each involving a different bond in the peptide backbone. Ionized peptides within a mass spectrometer carry charge in the form of a delocalized, mobile proton (Wysocki *et al.*, 2001) which can migrate along the length of the peptide backbone. Following a backbone fragmentation event, the charge (as a mobile proton) can remain on the resultant N-terminal or C-terminal fragment, as summarized in Figure 2.18. After the peptide bond is broken during CID, any charge carrying N-terminal fragments are termed *b*-ions and charge carrying C-terminal fragments are termed *y*-ions. Because mobile protons carry like charges, they Coulombically repel one another, and so tend to be distributed along the length of the peptide ion. For multiply charged precursor ions carrying multiple charges, this mobile proton behaviour favours the formation of fragment ions that carry a fraction of the precursor charge.

In the AnchorMS, fragmentation models only the *b* and *y* fragment ion series. The formation of ions in the *b* and *y* fragment ion series is more probable than those in other series generated by low energy CID. These *b* and *y* fragments are consequently detected with greater intensity (Barton *et al.*, 2007) and are typically prioritized for inclusion during library construction. Multiple fragmentation events are not supported. Ions resulting from such fragmentation occur in much lower abundance requiring two independent fragmentation events.

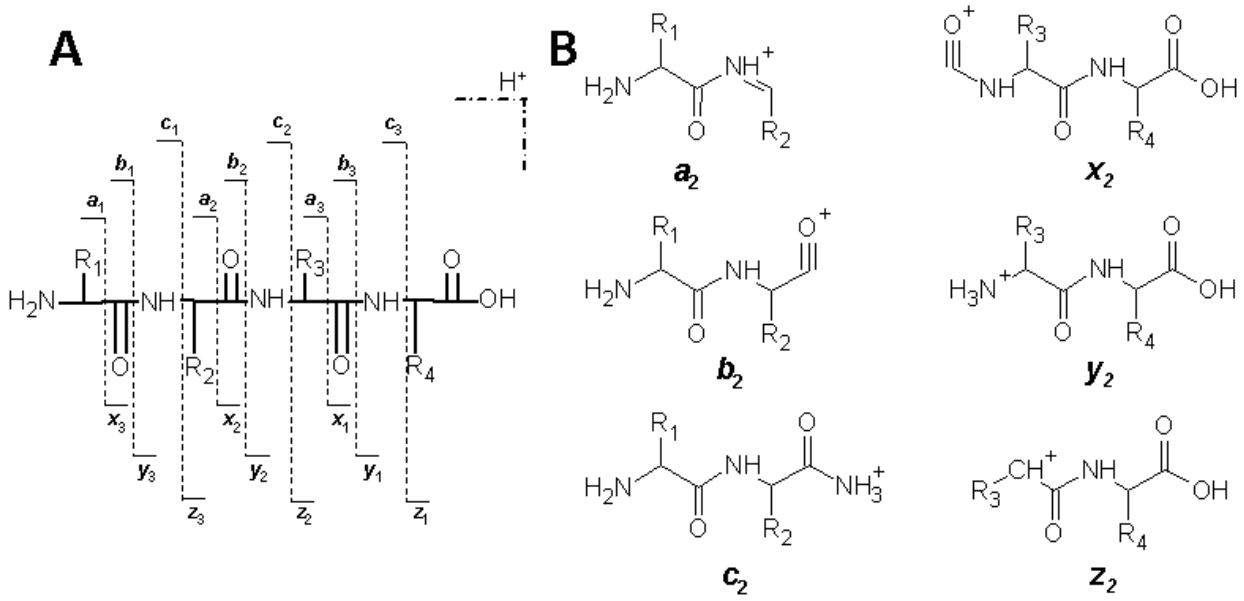


Figure 2.18: The diagram above illustrates CID fragmentation of peptides and shows the structural differences between six primary fragment types produced (Roepstorff and Fohlman, 1984), using a four-residue peptide as an example. (A) During low energy CID, peptide ions can fragment by 3 pathways. Each breaks a different bond and releases two fragment types. Fragments are named according to the bond broken and the number of residues in the fragment. Each vertical dotted line indicates a fragmentation site. The direction of the horizontal line indicates which fragment carries a charge and bears the adjacent fragment name. (B) The chemical structure of each fragment type, all products of low-energy CID fragmentation of the peptide backbone, is shown.

2.7.2.2.2 Mass calculations

During *in silico* fragmentation, each potential fragmentation site is traversed, and the mass of the peptide sequence on either side is calculated. These base masses are modified according to the chemical changes peculiar to each fragment ion type, as shown in Figure 2.18. Certain residues are also prone to the loss of H_2O, NH_3, CO or CO_2 during CID within the mass spectrometer. These possible losses are included (see Table 2.3), and the resultant species stored as distinct fragment ions. Fragments in the *a* series are thus also included since a loss of CO from a *b* fragment creates the equivalent of an *a* fragment. Where the susceptible residue is chemically modified, these chemical losses are not considered.

Formula	Monoisotopic molecular weight	Loss-sensitive residues in *b* ions	Loss-sensitive residues in *y* ions
NH_3	17.0265 Da	Q, N, K, R	Q, N, K, R
H_2O	18.0106 Da	S, T, D, E	None
CO	27.9949 Da	F	F

Table 2.3: Table of common chemical losses within the mass spectrometer during CID fragmentation (Papayannopoulos, 1995) implemented in AnchorMS.

Each modification in the theoretical di-peptide precursor is associated with a particular residue, and its mass must be added to the fragment containing that residue. As shown in Figure 2.18, CID dissociation also adds two hydrogen atoms to the fragment terminal adjacent to the dissociated bond in *c* and *y* ions (see Table 2.4).

Ion type	Chemical group added to peptide terminal	Chemical group added to fragment terminal	Change in monoisotopic mass of fragment	Change in average mass of fragment
a	H	(No change)	1.007825 Da	1.007976 Da
b	H	(No change)	1.007825 Da	1.007976 Da
c	H	H_2	3.023475 Da	3.023928 Da
x	OH	(No change)	17.00274 Da	17.00738 Da
y	OH	H_2	19.01839 Da	19.02333 Da
z	OH	(No change)	17.00274 Da	17.00738 Da

Table 2.4: The changes to the atomic chemical formulae of each low-energy CID fragment type.

2.7.2.2.3 Co-fragmentation and spectra overlap

Di-peptide precursors containing the same single-peptide sequences and cross-linker will be exactly the same mass (isobaric), and generate MS^1 peaks at the same *m/z* value, even if the cross-linking sites differ. However, if there are multiple residues reactive to the cross-linking reagent on either peptide within the di-peptide precursor, then either residue may be cross-linked, and instances of both formations may be observed. With different sites of cross-linking, each of these alternatively cross-linked di-peptides will also generate different fragmentation spectra (see Figure 2.19). The position of the cross-link will determine which fragment ions in a fragmentation series

carry the additional mass of the other, unfragmented peptide. Where basic residues are cross-linked (removing a chargeable site), the position of the cross-link significantly influences the fragmentation spectrum.

While the position of cross-linking sites may affect ionisation, especially if basic or acidic residues are cross-linked, it is a distinct possibility that two such isobaric molecules may have the same charge state and thus display identical *m/z* values. If this happens, and both are present in the same MS run, they may enter the instrument collision cell simultaneously. This would result in overlapping MS^2 spectra as depicted in Figure 2.19.

Typically MS3D software does not take this scenario into account. Certain upstream sample purification steps such as reverse phase liquid chromatography are capable of separating highly similar peptides (Cacia *et al.*, 1993; Chicz and Regnier, 1988; Chicz and Regnier, 1989; Frenz *et al.*, 1994) and, by extension, are likely also able to separate sequence-identical di-peptides with different cross-linked residues. In separating sequence-identical di-peptide precursors, each elutes separately and avoids co-fragmentation. In the absence of this experimental step or as a down-stream analysis safe-guard, the possibility of sequence-identical di-peptides can be considered during fragment library construction and MS^2 analysis.

To identify such occurrences of co-fragmentation, AnchorMS generates all possible combinations of the MS^2 spectra from the candidate assignments for each matched experimental precursor. The *get_permutations()* function in Permutations.py is used for this task in a similar fashion to the calculation of peptide modifications. This allows AnchorMS to recognize the presence of multiple di-peptide cross-link configurations in a mixture.

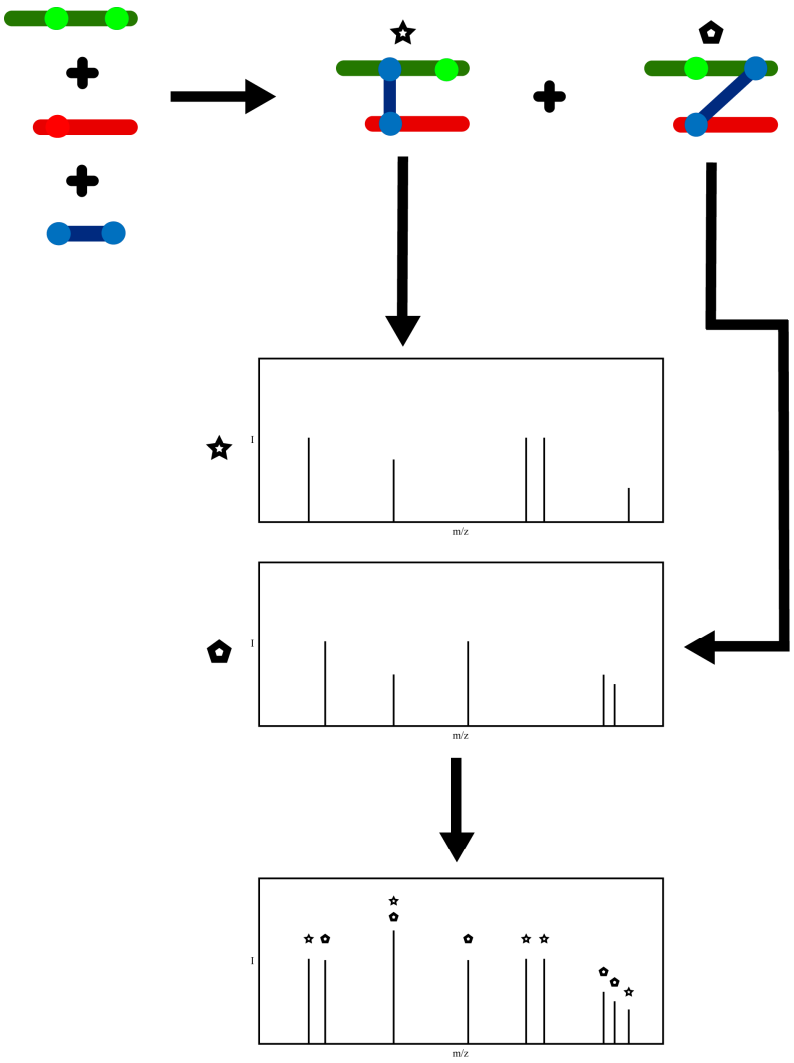


Figure 2.19: A diagram showing the co-fragmentation of isobaric precursors and the consequent overlap of their fragmentation spectra. The symbol above each peak in the bottom mass spectrum indicate which precursor that peak is derived from, and corresponds to one of the two mass spectra at the top of the diagram.

2.8. Matching MS^2 spectra and determining which residues are cross-linked

2.8.1 MS^2 matching within the AnchorMS workflow

During MS^1 analysis, AnchorMS identifies putative di-peptide precursor peaks within the experimental MS^1 spectrum. To confirm the identity of these matches, and to determine which residues are cross-linked, each putative di-peptide precursor is fragmented by CID. The resultant MS^2 spectra are compared against the predicted fragmentation spectra generated for each di-peptide that may be present in MS^1 spectrum. The similarity between each compared pair of predicted and observed MS^2 spectra is scored by AnchorMS.

2.8.2 Peak matching during MS^2 analysis

In contrast to MS^1 peak matching, MS^2 analysis with AnchorMS only considers a single match for each observed fragment peak in the experimental MS^2 spectrum. Any additional peaks within the matching tolerance of the observed fragment peak that match, are disregarded. This is because, in the MS^2 analysis, the overall similarity between MS^2 spectra is considered to be of greater importance than the specific identity of each fragment peak. In terms of MS^2 spectra matching, the AnchorMS scoring scheme considers only the number of matches. Thus, only a single match fragment is required for each experimental peak rather than a number of putative matches.

2.8.3 Module organisation

In AnchorMS, MS^2 matching is perform by *match_peaks()* in the Compare_Spectra.py module, the same function used for MS^1 peak matching. The m*atch_MSMS_spectra()* function in the same module is also involved in MS^2 matching, and directs the spectra comparisons performed by *match_peaks()*.

2.9. Scoring

As described in section 2.8 above, observed MS^2 spectra are matched against predicted MS^2 spectra for the putative di-peptide precursors identified during MS^1 analysis, to confirm the identity of putative di-peptide MS^1 peaks and to determine which residues are cross-linked. Each predicted MS^2 spectrum represents a distinct di-peptide ion, with its own specific peptide sequences, charge state, and cross-linked residues. The whole-spectrum matching of MS^2 spectra evaluates the closeness-of-fit between each pair of spectra compared. The predicted MS^2 spectrum which shows the closest match to a particular observed MS^2 spectrum, indicates the most likely identity and cross-linking sites for the precursor of that observed MS^2 spectrum.

2.9.1 Number of fragment peaks matched (*N*)

To identify which spectrum match is the closest, the closeness-of-fit between two spectra must be quantitated. Quantitative measures can be compared in a consistent fashion. In AnchorMS, the closeness-of-fit between each pair of compared spectra is measured in terms of the number of fragment peaks in the spectrum that are matched.

Where multiple di-peptide assignments were identified for an observed MS^1 peak, each

must be considered and the correct assignment determined through MS^2 analysis. After MS^2 spectrum matching, the candidate precursor assignments for a given observed MS^1 peak are ranked according to *N*, i.e. the number of fragment peaks matched between their predicted MS^2 spectrum and the observed MS^2 spectrum. The candidate di-peptide precursor that scores the highest number of fragment peak matches (*N*) is deemed the most likely assignment.

2.9.2 Unique Match Count (*UMC*)

2.9.2.1 N scores are not reliable for sequence-identical precursor

The *N* score is used to evaluate which candidate di-peptide assignment is the most likely. However, where two such candidate di-peptides have identical peptide sequences and differ only in the residues that are cross-linked, the predicted MS^2 spectra for each may be very similar, sharing a large number of identical fragment peaks. These shared fragment peaks contribute to the *N* score of each, and every identical peak that is matched in the one predicted spectrum, will also match in the other. Therefore, it is likely that the *N* score for sequence-identical precursor candidates would be very similar. As a result, there will not always be a good separation in *N* scores between correct and incorrect precursor assignments. The *N* score, therefore, cannot be considered reliable for discriminating between sequence-identical precursor candidates.

2.9.2.2 UMC scores discriminate between sequence-identical precursor candidates

To increase discrimination in such cases involving sequence-identical candidates, AnchorMS calculates a second score, specifically included for sequence-identical candidates. Even though sequence-identical precursors share many fragments, the remaining minority of fragments are distinct for a particular di-peptide precursor. For sequence-identical precursors, AnchorMS considers only those fragment peaks which are unique to the predicted MS^2 spectrum of each candidate precursor.

AnchorMS counts the number of these unique peaks which are matched during MS^2 spectrum matching. This value constitutes the second score in the AnchorMS scoring scheme, and is referred to as the *Unique Match Count* (UMC).

By considering only those peaks which are predicted to differ between the spectra of alternative candidate precursors, the *UMC* score is more discriminating than the *N*

score, and is not affected by high sequence-similarity.

2.9.2.3 Limits on the applicability of the UMC score

However, the *UMC* is only applicable to sequence-identical candidates because sequence-identical candidates will have the same number of and composition of residues, and thus will also have the same number of predicted fragment peaks. Precursor candidates that differ in their sequence, mass or charge state may also differ greatly in the number of fragment peaks in their predicted MS^2 spectra. A disparity between the number of peaks in the predicted spectra of alternative candidate precursors, adds an additional variable which may affect the *UMC* scores. For a larger precursor (at a higher charge state, to render its precursor peak value similar to that of the other candidate) with many unique peaks in its predicted MS^2 spectrum, a small number of peaks may match by chance. For the smaller candidate with fewer unique peaks in its predicted MS^2 spectrum, the likelihood of as many chance matches occurring is low.

The *UMC* score fulfils a specialized function within AnchorMS, as a second layer of discrimination in evaluating the quality of putative precursor assignments with similar *N* score values.

2.9.3 Number of fragment peaks falsely matched (*D*)

During an MS^2 analysis two questions must be answered for every putative di-peptide precursor match in the observed MS^1 spectrum. The first is, among the candidate di-peptide precursor assignments, which candidate predicted spectrum best fits the observed MS^2 spectrum and is thus the most likely correct assignment? This question is answered using the *N* score and *UMC* score, where the candidate with the highest *N* score or the highest *UMC* (for sequence-identical candidates) is called as the best candidate.

The second question to be answered, is whether all of the candidate di-peptide precursors are the incorrect assignment, arising from a chance false MS^1 match? Neither the *UMC* score, nor the *N* score on its own, can answer this second question. The *UMC* score only compares two alternative candidate precursors, not the validity of a single candidate.

Because the number of fragment matches (*N*) correlates with the closeness-of-fit between the observed and predicted MS^2 spectra, then in theory, a very low *N* score

would indicate a false precursor match. If all precursor assignment candidates had very low *N* scores, then none of them would be deemed incorrect assignments, and the MS^1 precursor match would be deemed a chance false positive match. However, in practice, one cannot be sure how low an *N* score needs to be to conclude that a spectrum match is spurious. In order to meaningfully interpret the *N* score value, the number of fragment peaks likely to be matched (false positive match) by chance, for an incorrect candidate precursor, needs to be estimated. This typical *N* score for a false positive match, we term the *D* score.

With an estimated *D* score, each *N* score could be compared to the estimated *D* score. If the *N* score is below the *D* score, then the candidate precursor is considered the incorrect assignment and discarded. Where the *N* score is close to the *D* score, the assignment is considered to be a low-confidence assignment.

In order to estimate the rate of false positive fragment peak matching, false positive matching or decoy matching were simulated *in silico* (see Chapter 3). A large number of theoretical di-peptide precursors were randomly generated and the sequence of each shuffled. For each such pair of precursors, the predicted MS^2 spectra for the original and sequence shuffled decoy precursor were matched against each other. The number of fragment peaks matched (*D*) was surveyed across all of the *in silico* matches. In Chapter 3, the data from this false matching simulation was used to derive equations for estimating the *D* score for a given theoretical di-peptide precursor, depending on the number of residues in its sequence, its precursor charge and the tolerance applied during matching. These equations are all incorporated into the AnchorMS scoring scheme.

2.10. Ranking and assignment

Throughout the processes of MS^2 matching, AnchorMS attempts to establish the identity of each putative di-peptide peak in the experimental MS^1 spectrum by matching its experimental MS^2 spectrum against the predicted MS^2 spectra of candidate theoretical di-peptides. Each of these fragment spectrum matches is assigned two scores, namely the number of fragment matches and the unique match count. The candidate assignment with the highest score is taken to be the assignment more likely to be correct.

However, for proximal precursor peaks in the experimental spectrum, the same candidate assignment may score highest for both experimental precursor peaks. It is unlikely that both observed peaks are exactly the same precursor ion. Therefore, one of the matches is taken to be a false positive.

Figure 2.20 shows such a scenario. Here, candidate D appears to be the leading contender for the identity of both precursor peak A and B. Because candidate D scored higher in its match to precursor peak B, AnchorMS considers it more likely that precursor peak B arises from candidate D, as opposed to precursor peak A. Therefore, candidate D is rejected as an assignment for precursor peak A. Disregarding candidate D, the next most likely assignment for precursor peak A is candidate G. In this way AnchorMS avoids assignment clashes in its presentation of the final results.

The ranked results of an AnchorMS analysis are displayed to the user through the AnchorMS web interface.

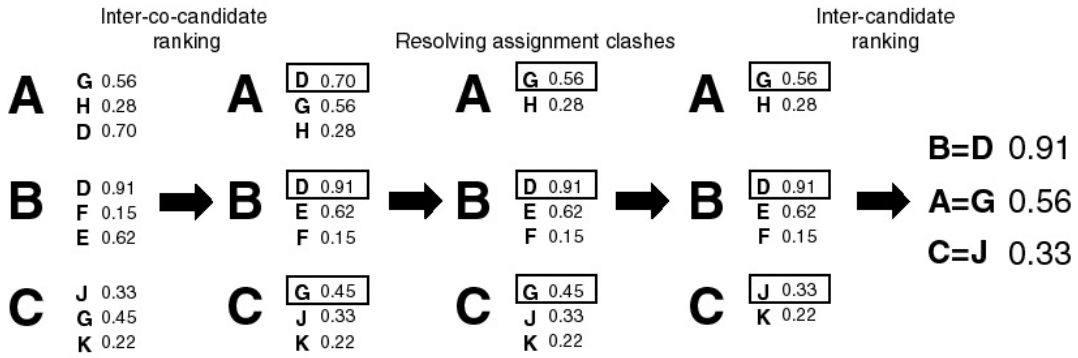


Figure 2.20: Summary of the ranking and assignment process using a simplified symbolic representation as an example. The larger letters to the left and the smaller letters to the right represent experimental spectra and theoretical candidate assignments, respectively. The leftmost column transition depicts candidate ranking. The central two column transitions depict the resolution of assignment clashes. The rightmost column depicts assignment ranking.

2.11. User interface and web-based front end

AnchorMS is intended primarily as a web service but command line input of data is also supported where AnchorMS is run locally. AnchorMS is open source and a Python module that may be downloaded for further development. Running AnchorMS as a web service allows updates and adjustments to be more readily and rapidly available to all users. Nonetheless, it is acknowledged that where large batch analyses are necessary, a locally run instance of AnchorMS may be more efficient. The web interface and associated front end functionality of AnchorMS is coded in XHTML and PHP, and runs on a local Apache web server (http://cbio.ufs.ac.za/AnchorMS).

If a given session involves a protracted computation time, then the results will not be ready immediately. When a requested analysis is completed, a link to the results page will be emailed to the user.

2.11.1 Web-based user interface

The AnchorMS interface is designed to be user-friendly with a simple layout. The workflow of the interface is similar to the experimental workflow of the user during sample preparation. Each step in the process is boxed and numbered. At the top of every page and alongside every section involving a distinct task, there is a link to specific help information regarding that page or task. The portal page contains a basic, introductory description of AnchorMS for the benefit of first time visitors to the website.

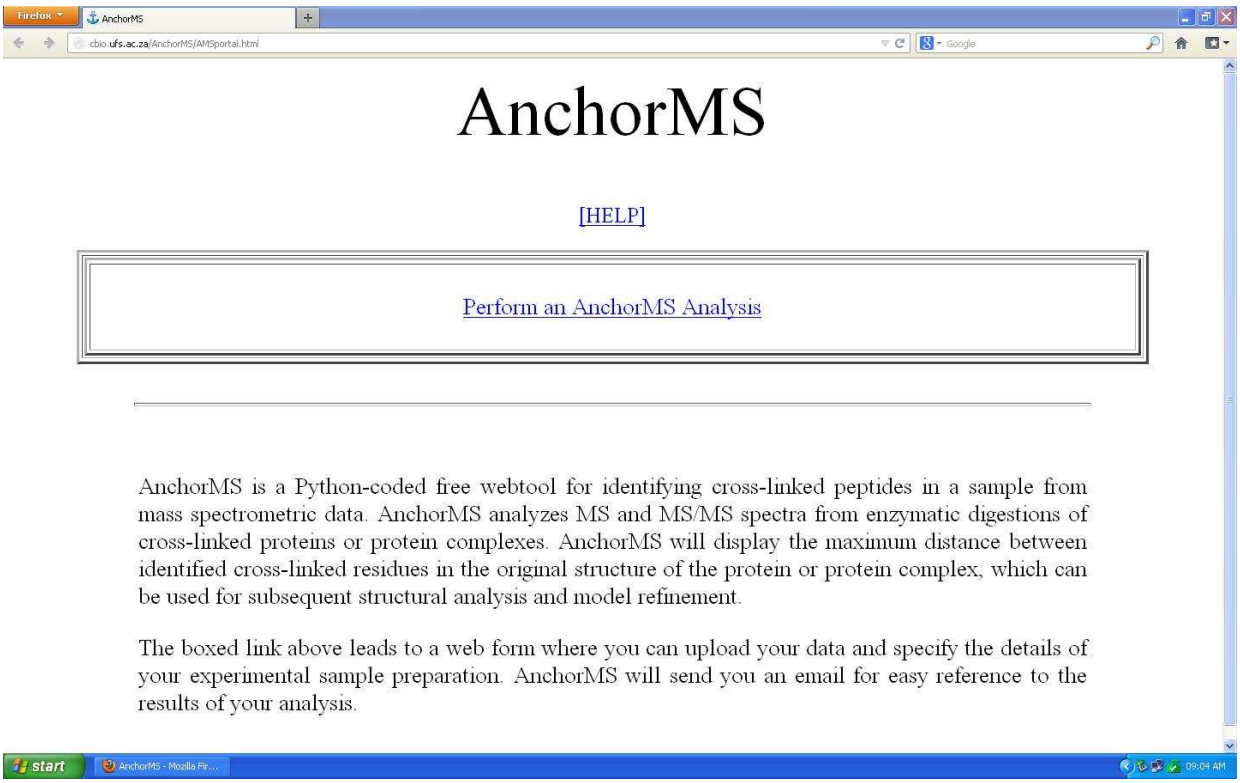


Figure 2.21: Screenshot of the AnchorMS portal page introducing visitors and users to the AnchorMS package, as implemented in the PHP/HTML script AMSportal.php.

From the portal page the user follows a link to a simple web form. Information pertaining to experimental conditions, treatments and samples are entered into the web form (see Figure 2.22). In total, the web form contains 8 separately boxed steps.

In step 1 the user is asked for protein sequence information in fastA format, either entered into a text box or uploaded as a text file. The fastA format simply requires each sequence to be preceded by a sequence name on its own line beginning with a ">" symbol. In step 2 the user must enter the mass accuracy of the instrument used, and choose between mono-isotopic and average mass values. Since average mass values are only applicable for low-resolution instruments, mono-isotopic mass values is the default selection. In step 3 the user selects which chemical losses are to be included in the simulation of CID fragmentation. "None" is selected by default.

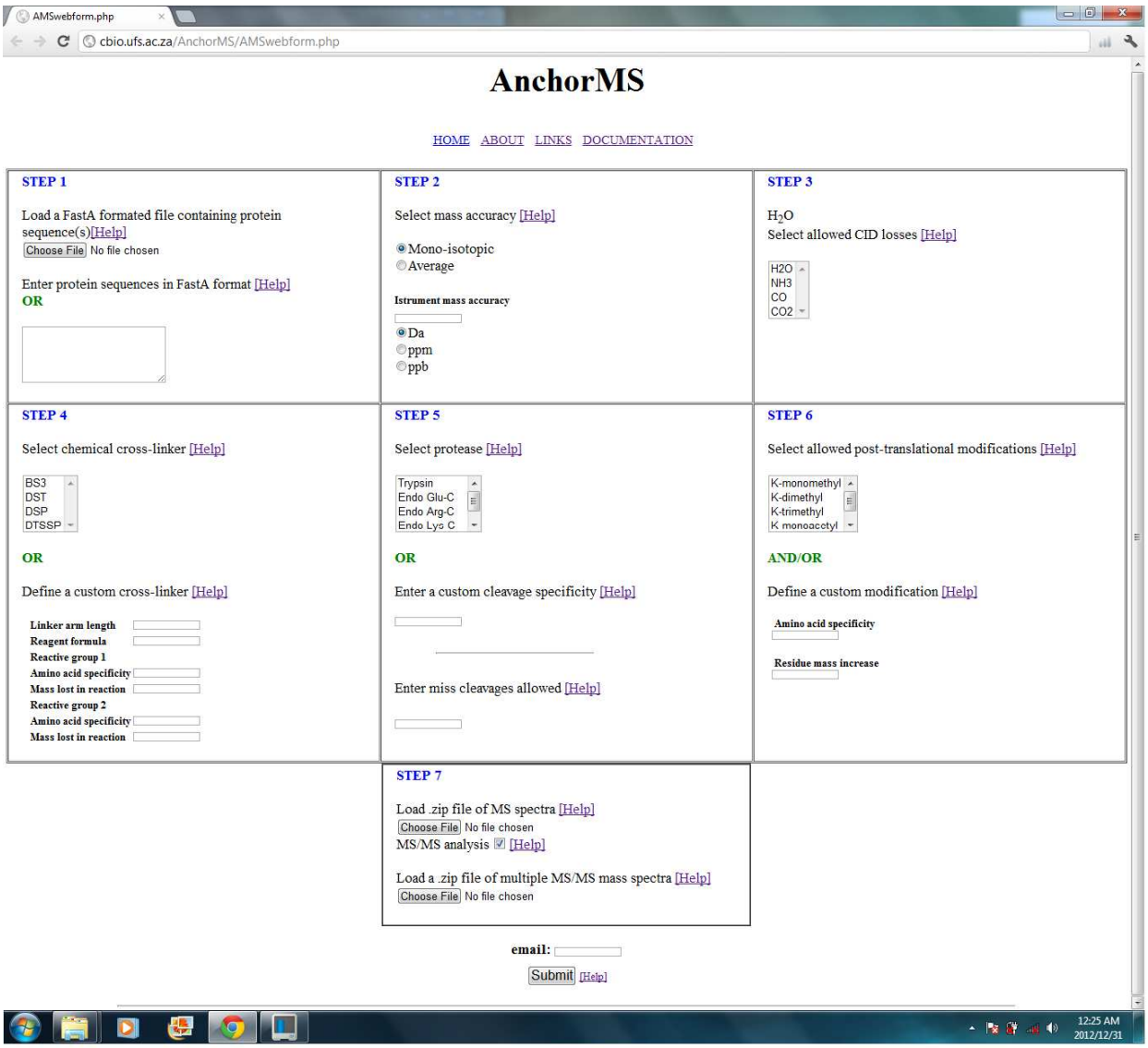


Figure 2.22: Screenshot of the web form which accepts user input for AnchorMS analysis as implemented in the PHP/HTML script AMSwebform.php.

In step 4 the cross-linking reagent(s) to which the sample was exposed is defined. Users can select any number of reagents from a list of known, commonly used cross-linker reagents. Alternatively, the user can define a custom cross-linking reagent by specifying the linker arm length and chemical formula of the reagent as well as the residue specificity and mass lost during the reaction for each reactive group on the cross-linking reagent. In step 5 the protease(s) with which the sample was digested is selected. Users can select any number of proteases from a list of known, commonly used proteases. Alternatively, the user can define a custom protease by specifying the sequence recognition for cleavage and the number of missed cleavages to be considered during the simulation of digestion. In step 6 those post-translational and induced residue modifications that are selected. Users can select any number of

modifications from a list of known modifications commonly observed or induced. Alternatively, the user can define a custom modification by specifying the mass change and residue specificity associated with the modification.

Steps 4 to 6 contain selection boxes for common cross-linking reagents, proteases and modifications respectively. The contents of these selection boxes are uploaded by PHP code within AMSwebform.php from three flat text files named “KnownCrosslinkers.txt”, “KnownProteases.txt” and “KnownModifications.txt”, respectively. Each line in these text files contains the name of a cross-linking reagent, protease or modification, and is displayed as a selection box entry. The entries in these three text files correspond exactly with the Python dictionaries ‘KnownCrosslinkers’, ‘KnownProteases’ and ‘KnownModifications’ in the Crosslinking.py, Digestion.py and Modifications.py modules in the Python back end, respectively.

In step 7 appropriate MS^1 and/or MS^2 data files can be uploaded to the web form. A checkbox, unchecked by default, can be checked if the user wishes to conduct only an MS^1 analysis. In an MS^1 only analysis, a sub-set of the MS^1 peaks uploaded are identified as potential di-peptide species.

In step 8 the user enters a valid email address. Users are notified by email when an analysis is complete and supplied with a link to the results. The results of prior AnchorMS analyses are kept on the AnchorMS server for a limited time that will depend on the server resources available.

2.11.2 Command line interface

If AnchorMS is accessed from the command line, the script "AnchorMS.py" is called, with first command line argument specifying a file path for a text file that contains cross-linking reagent information, protease information and various settings related to the AnchorMS analysis. Also in this text file is the file path of another fastA format text file containing the protein sequences. If this command line argument is absent then AnchorMS.py will default to looking for a file name “Configuration.py”. Figure 2.23 below shows example content of a valid settings text file. The file is structured as Python code, featuring a series of variable assignment statements.

```
MS_tol = 1.0
MSMS_tol = 0.01
MS_tol_unit = 'ppm'
MSMS_tol_unit = 'Da'
SinglePeptideMatch = False
SinglePeptideRank = False
SinglePeptideAssign = False
MSMSpeak_SingleMatching = True
MSMSpeak_ExclusiveMatching = False

MS_spectrum_data = "peak"      # element of ["mass", "peak"]
MSMS_spectrum_data = "peak"      # element of ["mass", "peak"]
masstype = 1            # element of [0, 1] --> 0:<isotopic average
                          mass>; 1:<monoisotopic mass>
max_charge = 6

miss_cleaves = 2
protease_info = ['K',1,'R',1]   # Trypsin: Cuts C-terminal to K & R
residues

crosslinker_info = [('E','D'),('K'),0.0,(-18.010565, -18.010565)]
       # EDC: Cross-links a E or D residue to a K residue, displacing H2O
       # and leaving a spacer arm of 0.0 Dalton

max_PTM_nmr = 2
fixed_mods = {'C': (57.021464,0), 'Y': ('SO3','H')}
# Acetylation of C adding ~57 Da & Sulfation of Y displacing 'H' with SO3
var_mods = {'K': ('CH3', 'H'), 'K': ('CH3', 'H')}

b_y_only = True
max_precursor_nmr = 2     # The maximum number of precursors that
                            can co-fragment

sequences_filename = 'proteinsubunits.fasta'
crosslinker_filename = 'CL_input_EDC.txt'
protease_filename = 'enzyme_input_trypsin.txt'
MS_filename = 'precursorscans.ms2'
MSMS_filename = 'productscans.ms2'
sample_title = 'MS3D_experiment_22'
```

Figure 2.23: Example lines from a valid input file for AnchorMS, which is created by either the user or the AnchorMS web interface, and uploaded to AnchorMS via the command line.

2.12 Discussion

AnchorMS was introduced as an application for effectively identifying di-peptide species in a sample based on MS^1 and MS^2 data, thereby fulfilling a central function in an MS3D workflow. In stepping through the simple and straightforward web interface, AnchorMS presents an easy-to-use application. A distinctive scoring scheme is implemented in AnchorMS which incorporates results from large-scale false matching simulations (see Chapter 3).

2.12.1 Coding style

The core functionality of AnchorMS has been written as a collection of Python routines. The web-based front end is coded entirely in PHP and XHTML. Chapter 3 will show that all calibration operations were scripted in R, a statistically-orientated language.

PHP was chosen for the web-related functionalities since it interfaces easily with Python and Apache. Python was favoured as the back end coding language being a versatile, open source language. In addition, a great many libraries for chemical and biological analyses are publicly available in Python. In this environment future extensions to and customization of AnchorMS are well facilitated and supported.

Despite the availability of various open source Python modules, the inclusion of external code was found to be unnecessary. Such in-house coding of ground level functionality by a single programmer is in fact conducive to overall coherence within the programme code. The coding of functionality which involves the reading and parsing of proprietary format data files, however, poses a special hurdle since the internal structure of the data file is not publicly available. In this case, the inclusion of external functionality in the form of the free '*MSconvert*' application provided a simple and effective solution.

At present collections of related Python dictionaries and lists form the heart of the theoretical library implemented in AnchorMS. Initial, preliminary versions of AnchorMS employed these library structures which threaded through the Python code, each having to be separately parsed into each Python function that made use of them. Because the order of parsed library elements is consistent throughout the code, this did not pose any particular difficulties. However, it became clear that future amendments and modifications to AnchorMS may have to handle more detail than was initially necessary. The current version of AnchorMS bundles these library structures together

into single Python objects. This greatly simplifies parsing within the function structure of AnchorMS. Additional, associated information that may support extended functionality can now be parsed into functions without adjusting the order of function parameters. Nonetheless, the current use of objects in AnchorMS does not fully and truly exploit object inheritance. Future versions of AnchorMS may embody a more fluid flow of information where objects inherit existing information rather than assigning selected information within the *__init__()* function. In addition to the objects used in AnchorMS, a sequence and peak object may prove to be a versatile construct. In essence, the *MatchPeaks()* function running in exclusive matching mode and the *AssignHighestScoringMatch()* method of the 'Assignment' object perform similar tasks. Future development with AnchorMS may explore the merging of these two processes into a single function that can be used in both Ranking.py and Compare_Spectra.py for spectrum matching and ranked assignment respectively.

2.12.2 Scoring

The scoring scheme of AnchorMS includes two features unique amongst available MS3D software: The 'unique match count' and the simulation-derived equation for estimating false positive matching scores.

With any scoring measure there is a chance of a false positive match. The user needs to have some sense of what score value is typical for false matches, and thereby gauge from the observed score value what the approximate likelihood of a given score value representing a false positive match is. AnchorMS assists in this process by providing a simulation calibrated baseline that describes the distribution of changes in expected false positive values under a specific set of conditions. This baseline model provides a frame of reference for the meaningful interpretation of the score value. Nonetheless, score values that are actual probability values, such as with MassMatrix (Xu *et al.*, 2008) or XLINK (Seebacher *et al.*, 2006), have the advantage that the level of inherent uncertainty in a match is incorporated into the value of the score. However, to calculate a precise probability of a given match being a true positive match depends upon the shape of the distribution of decoy score values being defined. In the case of AnchorMS, generalized equations that approximate the density distributions of decoy scores from the simulation data could be used as a basis for such probability distributions. Future versions of AnchorMS are likely to also include estimates of the probability of true positive matching for each putative hit.

2.12.3 Runtime

During AnchorMS development, the relative runtime required for various computational tasks was explored with a view to making AnchorMS faster. Several runtime bottlenecks were identified within the AnchorMS code and Python itself. Writing to or reading from text files in Python takes a relatively long time. Initially, AnchorMS stored identifier values for elements in its precursor and fragment libraries on file, which required repeated reading from the file on disk and slowed AnchorMS significantly. The introduction of an internal variable stored in memory circumvented this bottleneck.

Typically, in Python, when a copy is made, such as the assignment of the contents of one variable to the a second variable, no information is actually copied on memory. Instead the second variable name merely points to the original memory address. Consequently, any change to the value of the first variable will also affect the second variable. To avoid this and ensure that the contents of each variable is stored in separate memory addresses, the built-in Python function *deepcopy()* function can be used. However, the use of *deepcopy()* turned out to be a significant runtime bottleneck. Our function *DeepCopy()* was used for the same purpose, but was faster than the *deepcopy()* function.

Despite the large combinatorial space involved, the matching of MS^2 spectra collectively requires a great deal more time than library construction. This disparity could not be traced to an isolated procedure. In preliminary versions of AnchorMS the use of mySQL databases, accessed via the mySQLpy.py module, was compared with the use of internal Python data structures for housing the theoretical libraries. For the data sets used, little difference in performance was observed. However, since mySQL is specifically designed to handle large databases (Becla and Lim, 2008), had the size of the test data set been dramatically scaled up, differences may well have become evident.

Library construction and spectrum matching involve performing the same task for multiple categories. As such, this algorithm may enjoy significant gains from the use of threading, performing these tasks in parallel. The Python language does indeed support threading with both the Threads.py module and its successor, Threading.py. A prominent aspect of modern computing hardware is the inclusion of multiple processors even in desktop computers (Johansson *et al.*, 2000; March, 2000). Code designed to perform multiple computations in parallel would be better positioned to exploit this

architecture.

2.12.4 Matching

In practice, a cross-linking reaction typically proceeds with a low efficiency, producing a relatively small number of cross-linked di-peptides and a relatively large number of uncross-linked single peptides (Sinz, 2003). AnchorMS shortens its computation time by discarding all products that are not of interest. That is, single peptides, with and without attached cross-linker molecules, are not considered. Beyond indicating that the site is accessible to solvent molecules, internal and dead-end cross-linkages on a single peptide do not provide any structural information. Only di-peptides provide distance information necessary for MS3D. AnchorMS does not exclude potentially non-di-peptide peaks by identifying such peaks as peaks derived from molecules not of interest. Instead, AnchorMS assumes that the fragmentation spectra from such peaks will match poorly to those of theoretical di-peptides, and so be relegated to near the bottom of the assignment list with a low confidence score.

2.12.5 Fragmentation model

Although many of the principles and effects of CID fragmentation have been established, the underlying chemistry of CID fragmentation chemistry is not fully elucidated. In particular, there is a lack of accurate quantitative models that can predict the relative intensities of individual product ions (Bantscheff *et al.*, 2007). Nonetheless, some quantitative guidelines have been empirically established (Ji *et al.*, 2013; Paizs and Suhai, 2005) where the presence or absence of certain factors increase or decrease the intensity of specific fragment peaks (Kapp et al, 2003). The intensity values, however, still vary greatly (Barton and Whittaker, 2009). Arguably intensity information is best exploited for more accurate prediction by basing its interpretation on databases of experimentally derived fragmentation spectra, rather than on models. Software applications such as Mascot (Perkins *et al.*, 1999) make use of this approach. Publicly available experimental fragmentation spectra for cross-linked di-peptides are much scarcer than that of single peptides. As yet no freely accessible database of such di-peptide spectra has been established. With these factors in place the current version of AnchorMS does not rely on a database of experimental spectra, and does not use intensity information in its spectrum matching algorithm.

Despite the uncertainties involved, peaks with a higher intensity are less likely to be noise or contaminants. With this in mind, AnchorMS may well benefit from the

inclusion of intensity information in the scoring scheme. Fragment peak matches could be weighted according to the relative intensity of their peaks. For instance, the fractional number of matches could be multiplied by the sum of the relative intensities of all matched fragment peaks in a spectrum. With such a scheme in place, AnchorMS would favour assignments that match higher intensity fragment peaks.

In addition, a great deal of research is directed towards a better understanding of fragmentation chemistries. A number of sequence features appear to suppress or promote fragmentation or ionization at particular sites. For example, the proline effect denotes the increased observation of *y*-ions for fragmentation of the bond immediately N-terminal to a protonated proline residue (Paizs and Suhai, 2005; Schwartz and Bursey, 1992). Such sequence specific effects no doubt play an important role in the development of quantitative fragmentation models. Of special note for intensity independent spectrum matching in AnchorMS is the suppression of fragmentation at certain sites such that the fragment peak goes entirely unobserved. This incomplete sequence coverage can dramatically change observed MS^2 spectra, creating significant disparities with theoretical spectra. This illustrates the value of implementing a sequence sensitive fragmentation model. Many of the mostly widely used software packages for single peptide identification, such as SEQUEST (Yates, III *et al.*, 1995), assume uniform fragmentation across the peptide sequence. In the future, the collective set of observations connecting sequence features and peptide fragmentation behaviour will lead to theoretical MS^2 spectra being modelled in an increasingly sequence specific fashion. Of all the envisioned extensions to AnchorMS, the implementation of sequence specific fragmentation rules is the one likely to most improve the efficacy of AnchorMS since it renders each theoretical MS^2 spectrum more distinctive and narrows the discrepancy between the spectra modelled and those observed in practice.

In AnchorMS the number of matches (NoM) is normalized against the size of theoretical spectra as the fractional number of matches (NoMfr). With this in mind, low abundance fragmentation pathways were excluded from the AnchorMS fragmentation model. If fragments unlikely to be observed were to be considered, it would unduly increase the number of expected fragment peaks and unduly lower the NoMfr. The mobile ion model proposed by Wysocki and co-workers (Wysocki *et al.*, 2001) is a relatively recent addition to the understanding of how peptides ionize. In this model the ionization of peptides is conceived as being largely due to a delocalized charge rather

than necessarily being carried by an acidic or basic side chain. In light of the mobile ion hypothesis the current version of AnchorMS considers all fragment charge states below or equal to that of the precursor ion, regardless of the sequence of the fragment ion. That is to say, if, for example, the parent ion carries a charge of 4^+, then 1^+, 2^+, 3^+ and 4^+ are all given equal consideration as potential fragment charge states. While it may be theoretically possible to observe such charge states in light of the mobile ion model, other research suggests that acidic or basic side chains may affect ionization in peptide fragments (Biniossek and Schilling, 2012; Chawner *et al.*, 2012; Laskin *et al.*, 2012). Furthermore, if one considers the repulsive Coulombic forces between delocalized like charges, it seems unlikely that a high percentage of the delocalized charges on the precursor ion will cluster on one or the other fragment. Rather, it is more likely that the delocalized like charges on the precursor will divide between the fragment ions produced during CID. Thus, by considering even unlikely fragment charge states, AnchorMS unnecessarily increases the combinatorial space of the theoretical MS^2 library that is computed. Also, the inclusion of unlikely fragment charge states creates theoretical fragment spectra that are not realistic reflections of the fragment peaks that should be expected. This unduly lowers NoMfr values. In time, a better quantitative understanding of the factors involved in the retention of charges by post-CID peptides will emerge and can be exploited to improve future versions of AnchorMS.

The quadrupole of a mass spectrometer is designed to exploit differences in the mass to charge ratio between ions in order to separate chemically distinct molecules. This is very effective for single peptides since different peptide ions are very rarely exactly isobaric. The difference in mass and *m/z* value, even if small, can differentiate two species allowing separation by the mass spectrometer. Thus the possibility of multiple, differing precursors being co-fragmented has not featured much in the literature or available software. However, as described in section 2.6 of this Chapter, di-peptides can easily form precisely isobaric molecules that are indistinguishable in an MS^1 scan, and that may co-fragment. It is in fact fairly common to find multiple residues in a post-digest peptide that are reactive to the cross-linking reagent. This is especially true of classes of proteins where acidic or basic residues play a special role in the protein's function. For example, histone proteins are notably rich in lysine and arginine residues both of which are reactive to the popular cross-linking reagent BS^3. When considering two cross-linkable residues on a di-peptide, they are theoretically equally likely to be

cross-linked, expect in two cases: the first is where there is a significant difference in their distance from the cross-linked residue on the other strand. The second is where there is a difference in steric hindrance in the native protein conformation that may affect its accessibility to the cross-linking reagent.

It is possible that the co-fragmentation scenario is sometimes responsible for ambiguous results that suggest two possible cross-linking sites in an MS^2 spectrum. Because AnchorMS can consider co-fragmentation, it is particularly useful in resolving such ambiguities and revealing if co-fragmentation is a credible explanation. It is perhaps more realistic to consider an assignment that involves precursors with different cross-linking sites co-fragmenting.

2.13 Conclusion

AnchorMS has been designed with sufficient functionality to effectively identify di-peptide species as part of an MS3D experiment. While some judgement needs to be exercised by the user in its interpretation, the scoring measures generated by AnchorMS allow the user to assess the quality of displayed di-peptide matches. The back end computation is performed by coherent Python code divided into modules according to functionality. Independent of the Python functionality, a simple and straightforward web interface is used which also effectively manages concurrent job requests. A number of changes have been envisioned for future versions of AnchorMS, including a probabilistic score based on completed simulations, greater object orientation in the Python code, extensive use of threading for parallel computation, and a sequence specific dimension to the modelling of ionization and fragmentation of di-peptides. Since AnchorMS is released as an open source project, it is not only a contribution to the toolset available to researchers but also adds to the pool of code available to bioinformatics development.

2.14. References

1. Ambler, R. P. and Meadway, R. J. (1968) The use of thermolysin in amino acid sequence determination. *Biochem.J.* **108,** 893-895.

2. Bantscheff, M., Schirle, M., Sweetman, G., Rick, J., and Kuster, B. (2007) Quantitative mass spectrometry in proteomics: a critical review. *Analytical and bioanalytical chemistry* **389,** 1017-1031.

3. Barlow, C. K. and O'Hair, R. A. (2008) Gas-phase peptide fragmentation: how understanding the fundamentals provides a springboard to developing new chemistry and novel proteomic tools. *J.Mass Spectrom.* **43,** 1301-1319.

4. Barton, S. J., Richardson, S., Perkins, D. N., Bellahn, I., Bryant, T. N., and Whittaker, J. C. (2007) Using statistical models to identify factors that have a role in defining the abundance of ions produced by tandem MS. *Anal.Chem.* **79,** 5601-5607.

5. Barton, S. J. and Whittaker, J. C. (2009) Review of factors that influence the abundance of ions produced in a tandem mass spectrometer and statistical methods for discovering these factors. *Mass Spectrom.Rev.* **28,** 177-187.

6. Becla, J. and Lim, K. T. (2008) Report from the first Workshop on Extremely Large Databases. *Data Science Journal* **7,** 1-13.

7. Biniossek, M. L. and Schilling, O. (2012) Enhanced identification of peptides lacking basic residues by LC-ESI-MS/MS analysis of singly charged peptides. *Proteomics* **12,** 1303-1309.

8. Bonneil, E., Biringer, R., Saba, J., Huhmer, A., and Thibault, P. (2010) The Effect of Pressure Cycling on Proteolytic Cleavage Efficiency, Reaction Time and Protein Sequence Coverage. *peptides* **972,** 986.

9. Boyer, R. S. and Moore, J. S. (1977) A fast string searching algorithm. *Communications of the ACM* **20,** 762-772.

10. Bythell, B. J., Csonka, I. P., Suhai, S., Barofsky, D. F., and Paizs, B. (2010) Gas-phase structure and fragmentation pathways of singly protonated peptides with N-terminal arginine. *J.Phys.Chem.B* **114,** 15092-15105.

11. Cacia, J., Quan, C. P., Vasser, M., Sliwkowski, M. B., and Frenz, J. (1993) Protein sorting by high-performance liquid chromatography. I. Biomimetic interaction chromatography of recombinant human deoxyribonuclease I on polyionic stationary phases. *J.Chromatogr.* **634,** 229-239.

12. Chawner, R., Eyers, C. E., and Gaskell, S. J. (2012) The influence of a C-terminal basic residue on peptide fragmentation pathways. *International Journal of Mass Spectrometry*

13. Chen, Z. A., Jawhari, A., Fischer, L., Buchen, C., Tahir, S., Kamenski, T., Rasmussen, M., Lariviere, L., Bukowski-Wills, J. C., and Nilges, M. (2010) Architecture of the RNA polymerase IIâ€“TFIIF complex revealed by cross-linking and mass spectrometry. *The EMBO journal* **29,** 717-726.

14. Chicz, R. M. and Regnier, F. E. (1988) Surface-mediated retention effects of subtilisin site-specific variants in cation-exchange chromatography. *J.Chromatogr.* **443,** 193-203.

15. Chicz, R. M. and Regnier, F. E. (1989) Single amino acid contributions to protein retention in cation-exchange chromatography: resolution of genetically engineered subtilisin variants. *Anal.Chem.* **61,** 2059-2066.

16. Deutsch, E. (2008) mzML: a single, unifying data format for mass spectrometer output. *Proteomics.* **8,** 2776-2777.

17. Deutsch, E. W. (2010) Mass spectrometer output file format mzML. *Methods Mol.Biol.* **604,** 319-331.

18. Dobo, A. and Kaltashov, I. A. (2001) Detection of multiple protein conformational ensembles in solution via deconvolution of charge-state distributions in ESI MS. *Analytical chemistry* **73,** 4763-4773.

19. Frenz, J., Quan, C. P., Cacia, J., Democko, C., Bridenbaugh, R., and McNerney, T. (1994) Protein sorting by high performance liquid chromatography. 2. Separation of isophosphorylates of recombinant human DNase I on a polyethylenimine column. *Anal.Chem.* **66,** 335-340.

20. Horspool, R. N. (1980) Practical fast searching in strings. *Software: Practice and*

Experience **10,** 501-506.

21. Hume, A. and Sunday, D. (1991) Fast string searching. *Software: Practice and Experience* **21,** 1221-1248.

22. Jaitly, N., Mayampurath, A., Littlefield, K., Adkins, J. N., Anderson, G. A., and Smith, R. D. (2009) Decon2LS: An open-source software package for automated processing and visualization of high resolution mass spectrometry data. *BMC.Bioinformatics.* **10,** 87.

23. Ji, C., Arnold, R. J., Sokoloski, K. J., Hardy, R. W., Tang, H., and Radivojac, P. (2013) Extending the coverage of spectral libraries: A neighborhood based approach to predicting intensities of peptide fragmentation spectra. *Proteomics*

24. Johansson, J.M., March, S.T., and Naumann, J.D. The effects of parallel processing on update response time in distributed database design. 187-196. 2000. Association for Information Systems.
Ref Type: Conference Proceeding

25. Kallos, G. J., Tomalia, D. A., Hedstrand, D. M., Lewis, S., and Zhou, J. (2005) Molecular weight determination of a polyamidoamine starburst polymer by electrospray ionization mass spectrometry. *Rapid communications in mass spectrometry* **5,** 383-386.

26. Kessner, D., Chambers, M., Burke, R., Agus, D., and Mallick, P. (2008) ProteoWizard: open source software for rapid proteomics tools development. *Bioinformatics* **24,** 2534-2536.

27. Laskin, J., Kong, R. P., Song, T., and Chu, I. K. (2012) Effect of the Basic Residue on the Energetics and Dynamics of Dissociation of Phosphopeptides. *International Journal of Mass Spectrometry*

28. Lee, K. K., Sardiu, M. E., Swanson, S. K., Gilmore, J. M., Torok, M., Grant, P. A., Florens, L., Workman, J. L., and Washburn, M. P. (2011) Combinatorial depletion analysis to assemble the network architecture of the SAGA and ADA chromatin remodeling complexes. *Mol.Syst.Biol.* **7:503.,** 503.

29. March, S. (2000) Reflections on computer science and information systems

research. *Conceptual Modelingâ€”ER 2000* 105-181.

30. Martens, L., Chambers, M., Sturm, M., Kessner, D., Levander, F., Shofstahl, J., Tang, W. H., Rompp, A., Neumann, S., Pizarro, A. D., Montecchi-Palazzi, L., Tasman, N., Coleman, M., Reisinger, F., Souda, P., Hermjakob, H., Binz, P. A., and Deutsch, E. W. (2011) mzML--a community standard for mass spectrometry data. *Mol.Cell Proteomics.* **10,** R110.

31. Mathews, C. K., Van Holde, K. E., and Ahern, K. G. *Biochemistry*, Benjamin Cummings,2000.

32. Mayne, S. L. and Patterton, H. G. (2011) Bioinformatics tools for the structural elucidation of multi-subunit protein complexes by mass spectrometric analysis of protein-protein cross-links. *Brief.Bioinform.* **12,** 660-671.

33. McDonald, W. H., Tabb, D. L., Sadygov, R. G., MacCoss, M. J., Venable, J., Graumann, J., Johnson, J. R., Cociorva, D., and Yates, J. R., III (2004) MS1, MS2, and SQT-three unified, compact, and easily parsed file formats for the storage of shotgun proteomic spectra and identifications. *Rapid Commun.Mass Spectrom.* **18,** 2162-2168.

34. Mo, F., Mo, Q., Chen, Y., Goodlett, D. R., Hood, L., Omenn, G. S., Li, S., and Lin, B. (2010) WaveletQuant, an improved quantification software based on wavelet signal threshold de-noising for labeled quantitative proteomic analysis. *BMC.Bioinformatics.* **11,** 219-11.

35. Morohashi, M., Shimizu, K., Ohashi, Y., Abe, J., Mori, H., Tomita, M., and Soga, T. (2007) P-BOSS: a new filtering method for treasure hunting in metabolomics. *J.Chromatogr.A.* **1159,** 142-148.

36. Olsen, J. V., Ong, S. E., and Mann, M. (2004) Trypsin cleaves exclusively C-terminal to arginine and lysine residues. *Mol.Cell Proteomics* **3,** 608-614.

37. Orchard, S., Hermjakob, H., Taylor, C. F., Potthast, F., Jones, P., Zhu, W., Julian, R. K., Jr., and Apweiler, R. (2005) Further steps in standardisation. Report of the second annual Proteomics Standards Initiative Spring Workshop (Siena, Italy 17-20th April 2005). *Proteomics* **5,** 3552-3555.

38. Paizs, B. and Suhai, S. (2005) Fragmentation pathways of protonated peptides. *Mass Spectrom.Rev.* **24,** 508-548.

39. Papayannopoulos, I. A. (1995) The interpretation of collision-induced dissociation tandem mass spectra of peptides. *Mass Spectrometry Reviews* **14,** 49-73.

40. Pedrioli, P. G., Eng, J. K., Hubley, R., Vogelzang, M., Deutsch, E. W., Raught, B., Pratt, B., Nilsson, E., Angeletti, R. H., Apweiler, R., Cheung, K., Costello, C. E., Hermjakob, H., Huang, S., Julian, R. K., Kapp, E., McComb, M. E., Oliver, S. G., Omenn, G., Paton, N. W., Simpson, R., Smith, R., Taylor, C. F., Zhu, W., and Aebersold, R. (2004) A common open representation of mass spectrometry data and its application to proteomics research. *Nat.Biotechnol.* **22,** 1459-1466.

41. Perkins, D. N., Pappin, D. J., Creasy, D. M., and Cottrell, J. S. (1999) Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis* **20,** 3551-3567.

42. Rawlings, N. D. and Barrett, A. J. (2000) MEROPS: the peptidase database. *Nucleic Acids Research* **28,** 323-325.

43. Rawlings, N. D. and Salvesen, G. *Handbook of proteolytic enzymes*, Academic Press,2012.

44. Reinhold, B. B. and Reinhold, V. N. (1992) Electrospray ionization mass spectrometry: Deconvolution by an entropy-based algorithm. *Journal of the American Society for Mass Spectrometry* **3,** 207-215.

45. Roepstorff, P. and Fohlman, J. (1984) Proposal for a common nomenclature for sequence ions in mass spectra of peptides. *Biomed.Mass Spectrom.* **11,** 601.

46. Schrader, N., Stelter, P., Flemming, D., Kunze, R., Hurt, E., and Vetter, I. R. (2008) Structural basis of the nic96 subcomplex organization in the nuclear pore channel. *Mol.Cell.* **29,** 46-55.

47. Schwartz, B. L. and Bursey, M. M. (1992) Some proline substituent effects in the tandem mass spectrum of protonated pentaalanine. *Biol.Mass Spectrom.* **21,** 92-96.

48. Schwieters, C. D. and Clore, G. M. (2001) The VMD-XPLOR visualization package for NMR structure refinement. *J.Magn Reson.* **149,** 239-244.

49. Seebacher, J., Mallick, P., Zhang, N., Eddes, J. S., Aebersold, R., and Gelb, M. H. (2006) Protein cross-linking analysis using mass spectrometry, isotope-coded cross-linkers, and integrated computational data processing. *J.Proteome.Res.* **5,** 2270-2282.

50. Sinz, A. (2003) Chemical cross-linking and mass spectrometry for mapping three-dimensional structures of proteins and protein complexes. *J.Mass Spectrom.* **38,** 1225-1237.

51. Stromback, L., Hall, D., and Lambrix, P. (2007) A review of standards for data exchange within systems biology. *Proteomics.* **7,** 857-867.

52. Taverner, T., Hall, N. E., O'Hair, R. A., and Simpson, R. J. (2002) Characterization of an antagonist interleukin-6 dimer by stable isotope labeling, cross-linking, and mass spectrometry. *J.Biol.Chem.* **277,** 46487-46492.

53. Waugh, D. S. (2011) An overview of enzymatic reagents for the removal of affinity tags. *Protein Expr.Purif.* **80,** 283-293.

54. Wenig, P. and Odermatt, J. (2010) OpenChrom: a cross-platform open source software for the mass spectrometric analysis of chromatographic data. *BMC Bioinformatics* **11,** 405.

55. Wieser, M. E. and Coplen, T. B. (2011) Atomic weights of the elements 2009 (IUPAC Technical Report). *Pure Appl.Chem.* **83,** 359-396.

56. Wysocki, V. H., Tsaprailis, G., Smith, L. L., and Breci, L. A. (2001) Mobile and localized protons: a framework for understanding peptide dissociation. *J.Mass Spectrom.* **35,** 1399-1406.

57. Xu, H., Zhang, L., and Freitas, M. A. (2008) Identification and characterization of disulfide bonds in proteins and peptides from tandem MS data by use of the MassMatrix MS/MS search engine. *J.Proteome.Res.* **7,** 138-144.

58. Yates, J. R., III, Eng, J. K., McCormack, A. L., and Schieltz, D. (1995) Method to correlate tandem mass spectra of modified peptides to amino acid sequences in the protein database. *Anal.Chem.* **67,** 1426-1436.

59. Zhang, Z. and Marshall, A. G. (1998) A universal algorithm for fast and automated

charge state deconvolution of electrospray mass-to-charge ratio spectra. *Journal of the American Society for Mass Spectrometry* **9,** 225-233.

Chapter 3
A mathematical model to predict false positives in AnchorMS

3.1 Introduction

The AnchorMS software package was developed to identify cross-linked di-peptides by the analysis of MS^1 and MS^2 data. Based on the protein sequence and upstream sample preparation, AnchorMS generates a library of theoretical precursors and fragments, which may be observed in the experimental MS^1 and MS^2 spectra. To identify observed experimental peaks, such as those peaks produced by di-peptide species, AnchorMS compares the observed peaks against the predicted peaks in a generated theoretical library. However, occasionally an observed experimental peak is identified as being produced by a di-peptide when, in fact, it is not. This may occur when the observed peak value is similar to the peak value of a predicted peak in the theoretical library. An important aspect of MS-analysing software such as AnchorMS is how it minimizes these false positive matches. The matching of predicted and observed spectra with a high degree of stringency may reduce the number of false positive matches but, if excessively stringent, may simultaneously exclude correctly matched peaks, and so increase the number of false negative matches.

There are two kinds of false positive matches possible that need to be considered. In the first, a non-di-peptide species produces, by pure chance, a peak with a similar peak value to that of a predicted di-peptide peak. AnchorMS aims to exclude these peak matches. In the second, an observed peak is matched to multiple predicted peaks in the theoretical library. Here, AnchorMS must simply choose which of these matches is the most likely to represent the true identity of the observed peak. In order to exclude the first and to favour the correct assignment in the case of the second, the quality of each peak match must be quantitated.

In AnchorMS, for MS^1 peaks identified as putative di-peptides, the MS^2 spectrum of the MS^1 peak is matched against the predicted MS^2 spectrum. Matching whole MS^2 spectra in this way is more sensitive to false positive matches, because more peak values are involved in the comparison. AnchorMS assesses the quality of a spectrum match and the likelihood of a putative MS^1 peak match being correct, based on the number of fragment peaks matched in the MS^2 spectrum. The greater the number of

observed fragment peaks found to be consistent with the predicted MS^2 spectrum, the more reliable that match is considered to be. Thus, the number of fragment peaks matched serves as a "score value" with which the quality of spectrum matches can be quantitated.

In order to assess whether a quantitative score is sufficiently high for a confident match assignment, one must be able to define what is meant by a “high” or “good” score and what is meant by a “low” or “bad” score. The reference score value, deemed typical for false positive matches, serves as a threshold for evaluating other score values. Simply put, any score that is greater than this false positive threshold is assumed to represent a true positive match. Raising this threshold or baseline value will increase the stringency, and render matching more exclusive. However, in doing so, though the number of false positives will decrease, the number of false negative matches may concurrently increase as well. In effect, a high threshold score value favours sensitivity, even at the risk of excluding some di-peptide species.

To set a threshold value that optimizes the balance between stringency and including false positives is not always straightforward. One approach is to base the threshold score value on empirical observations of false positive matches. In this way the threshold value finds the right balance between accuracy and sensitivity by reflecting in its value a realistic estimation of the score value that is typical for a false positive mass. However, the score value that is typical for false positive matches may vary depending on conditions. If this is the case, then the choice of a fixed score threshold will only reflect a realistic estimation of typical score values for false-positive matches for a specific and applicable set of circumstances. To accommodate a variety of circumstances and conditions, the threshold or baseline can be defined as a dynamic threshold rather than a fixed value. Here, an equation that defines and calculates the threshold value, replaces the fixed threshold value.

AnchorMS employs a dynamic false positive threshold. A set of equations calculates a new threshold value for each di-peptide precursor considered. In this chapter, we derive a set of equations that define a dynamic false positive threshold for score values. To derive a dynamic false positive threshold, appropriate datasets are required. As detailed later in this Chapter, we generated such false matching sets by simulating false-positive fragment spectrum matching *in silico*. Di-peptide sequences were randomly generated and a second, decoy di-peptide precursor sequence was

generated by sequence-shuffling the first di-peptide sequence. The theoretical MS^2 spectrum of each was calculated and matched against each other.

AnchorMS also calculates a second score value, specifically in cases where multiple theoretical MS^1 peaks are putatively matched to a single observed MS^1 peak. This secondary score is dubbed the 'Unique Match Count' (*UMC*). A dynamic, false positive threshold is calculated for the *UMC* score as well.

In this chapter we investigate the level of false positive matching observed under various theoretical conditions, and examine the impact of each on both kinds of score values. Precursor length, precursor charge, tolerance and the distance between alternative cross-linking sites were identified as applicable factors.

We present the results of various subsets of the simulation data, and follow the step-by-step derivation of the decoy threshold equations. In each case we attempt to fit suitable models to the available data. A set of equations to quantitate the false positive or decoy threshold of AnchorMS was constructed by combining each derived model. These final threshold equations were incorporated into the Scoring.py module of the AnchorMS code.

3.2 Materials and Methods

The simulation of decoy matching was performed in a series of trials by a collection of similar Python scripts. The datasets produced were subsequently processed and summarized by a separate collection of similar Python scripts. This involved the creation of supersets and subsets of the generated data as well as various calculations therewith. The results of this processing were imported into a spreadsheet display (Microsoft Excel) for preliminary graphical visualization of data trends as well as to facilitate rudimentary manual manipulation of data and its arrangement. Preliminary manual adjustments and calibration of various model components were performed in Microsoft Excel. The results there from provided initializing parameter values for automated calibration performed by the ‘Calibration.r’ script written in the R programming language. Density plots of various data subsets were also generated using R code.

3.2.1 Decoy-Ideal Matching Trials

In a series of successive simulations, peptide sequences of varying length and amino acid composition were randomly generated and ideal spectra derived from each. These simulations can be grouped into two distinct trials.

The first trial simulated decoy-ideal matching in sequence-differing di-peptides. For each sequence, a decoy sequence was constructed by randomly shuffling the initial sequence three times and matching its theoretical spectrum against that of the initial sequence. As a measure of ideal spectrum matching, the ideal spectrum was also matched against itself. For each matching event, the number of matches, the number of peaks in the predicted spectrum (theoretical spectrum size), the precursor length, the precursor charge, the matching tolerance, the matching tolerance unit, and the unique match count were written to an archive text file. The basic workflow of these simulations is represented in Figure 3.1 and was implemented in the Python scripts bearing the filename tag “DecoyIdeal_Trial”.

In the second trial, decoy-ideal matching of sequence-identical di-peptides was simulated by a different script. Here the workflow was identical to that of the first trial, as depicted in Figure 3.1, with two exceptions: In the decoy di-peptide, the sequence was not shuffled and the cross-linked site was separately and randomly chosen. The sum of the distance between alternative cross-linked sites on each strand was calculated and added to the data written to the archive text file, mentioned above.

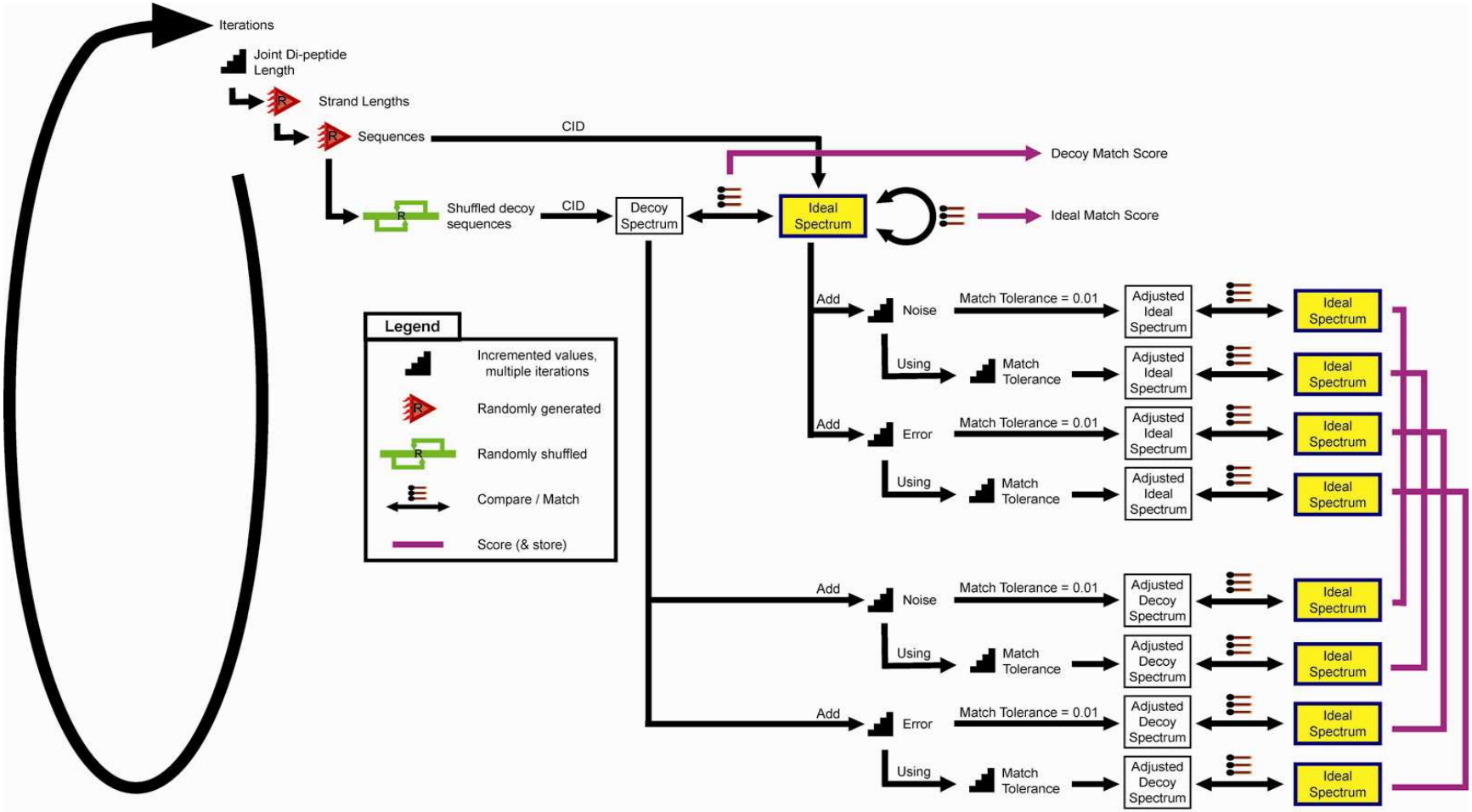


Figure 3.1: Workflow diagram for calculations in the decoy-ideal matching trials with sequence-differing precursors. The script iterates through joint di-peptide length values of between 3 and 50 residues. A peptide length (less than the joint di-peptide length) is randomly generated. The length of the other peptide is the difference between these two values. Amino acid sequences of the determined lengths are randomly generated for each peptide. Two decoy peptide sequences are generated by randomly shuffling the concatenated sequences and dividing it into two at a random site. Ideal and decoy theoretical CID fragmentation spectra were generated from the sequence and decoy sequence, respectively. The ideal spectrum as well as the decoy spectrum is compared with the ideal spectrum. The script iterates through precursor charge states, generating an adjusted ideal and adjusted decoy spectrum for each. Each adjusted spectrum is matched against the original ideal spectrum using various match tolerance values. Relevant matching measures for equivalent ideal and decoy-adjusted spectra were stored. This entire process is repeated over 1000 iterations.

3.2.2 DecoyIdeal_Extraction.py

The ‘DecoyIdeal_Extraction.py’ script was used to extract pertinent information from the raw data generated by Decoy-Ideal trial simulations, and stored in archive text files. The script accepted value ranges for all of the parameters recorded in the archive text files. These value ranges defines a subset of the simulation data which the script extracts and summarizes. For example, 'DecoyIdeal_Extractions.py' can be directed to only extract information for di-peptides with a precursor length of between 20 and 30 residues and a precursor charge of 2. Each decoy-ideal simulation run, and its related archive text file, is associated with a particular numerical identifier, which must also be specified for 'DecoyIdeal_Extractions.py'. These criteria define a subset of the simulation data. The script traverses the archive text file and checks whether each entry fits the subset criteria. Selected values within the criteria-conforming entries are stored within internal, multi-dimensional Python dictionaries. These dictionaries effectively group data entries that share the specific parameter values. In each such grouping the maximum, minimum, mean and standard deviation values are calculated. A selection of these values is written to text files summarizing the defined data subset.

3.2.3 Peptide_Difference_Survey.py

The ‘Peptide_Difference_Survey.py’ Python script randomly generates peptide sequence pairs, each containing a specific number of residues. For each sequence pair, the difference between their molecular weights is calculated and stored in a Python list. The values in this list are sorted and grouped into bins corresponding to value ranges of 0.02 Da using the custom *BoxingCenterDensity()* function. The mid-point value of each bin, as well as the number of data points contained therein, is written to text file. This process was repeated 1,000,000 times for each peptide sequence length (number of residues) within a defined length range. The rudimentary “density” data was subsequently used to generate appropriate graphs using either R or Microsoft Excel.

3.2.4 Calibration.r

The ‘Calibration.r’ script was coded in R and made use of the standard *nls()* function which fits non-linear models to data. Figure 3.2 details a generalized form of the key lines of code employed in ‘Calibration.r’. Within the script, variations of this code are repeated for each of the sub-models calibrated in this chapter. These sub-models together form the composite decoy matching baseline model implemented in AnchorMS.

```
1 Model <- function(var1, var2, parameter1, parameter2, parameter3) (<Model_Equation>)
2 rawdataset <- read.table(file = rawdata_filename, header = TRUE)
3 plotdataset <- rawdataset[which(rawdataset[, index1] & rawdataset[, index2]), ]
4 CalibData <- data.frame(x = plotdata[["<independent var>"]], y = plotdata[["<dependent
  var>"]])
5 CalibFit <- nls(y ~ I(Model(var1, var2, parameter1, parameter2, parameter3)), data =
  CalibData,
6            start = list(parameter1 = par1.seed, parameter2 = par2.seed, parameter3 =
             par3.seed), trace=T)
7 CalibFit_Summary <- summary(CalibFit)
8 write.table(CalibFit_Summary["parameters"], output_filepath)
```

Figure 3.2: The key lines of R code within the 'Calibration.r' script, adjusted to include generalized parameter names, are shown. The hypothesized model is defined (line 1). The raw data is read from file (line2) and its plot-salient subset extracted (lines 3-4). The model is fitted to this data using the *nls()* function and the parameter values optimized (lines 5-6). The parameter portion of the process summary is written to file (lines 7-8).

3.2.5 Density.r

The 'Density.r' script was coded in R and makes use of the *density()* function within the standard R library. This script was used to generate density plots for the normalized number of decoy matches in the sequence-differing trial as well as for Unique Match Count from the sequence-identical trial (see Figure 3.3 below).

```
1 rawdataset <- read.table(file=rawdata_filepath, header=TRUE)
2 plotdataset <- rawdataset[which(rawdataset[,index1] & rawdataset[,index2]),]
3 mindatapoint <- min(plotdataset[["<datalabel>"]])
4 maxdatapoint <- max(plotdataset[["<datalabel>"]])
5 meandatapoint <- mean(plotdataset[["<datalabel>"]])
6 densdata <- as.vector(as.matrix(plotdataset[["<datalabel>"]]))
7 sd.densdata <- sd(densdata)
8 densdistr <- density(densdata,kern="gaussian",from=mindatapoint,to=maxdatapoint)
9 dens.dataframe <- data.frame(x=densdistr$x, y=densdistr$y)
10 png(graphic_filename)
11 plot(densdistr, main=graph_title_string)
12 points(line.vector, (as.vector(1:length(line.vector))*scaling.factor), col="<colour>",
   pch=20)
```

Figure 3.3: The key lines of R code within 'Density.r', adjusted to include generalized parameter names, are shown. The raw data is read from file (line1) and its plot-salient subset extracted (line 2). The minimum, maximum, mean and standard deviation values are determined (lines 3-5 and 7). The density of dependent data values is calculated using the *density()* function and formatted into a dataset with a domain stretching from the minimum to maximum dependent data values (line 8). Lines 6 and 9 perform appropriate data format conversions. A .png graphics file is created (line 10). The density dataset is plotted against the dependent data values and written to file (line 11). Vertical "lines" of different colours are added to the plot on file to indicate minimum, maximum, mean and standard deviation values (line 12) and the graphic on file is closed (line 13). The "line.vector" object in line 12 is a multi-point, constructed dataset which approximates a solid line.

3.3 Results

In this study we set out to estimate the average level of false peak matching likely to occur during the matching of theoretical and observed fragmentation mass spectra. In order to investigate the factors that affect false matching levels, false matching was simulated *in silico* by constructing theoretical MS^2 spectra of randomly generated di-peptides and matching each against the theoretical MS^2 spectrum of a sequence-shuffled decoy di-peptide. The number of fragment peaks decoy matched (*D*) in this way was used as a measure of the level of false matching. In addition, to establishing how various factors affect the value of *D*, we sought to derive a mathematical model that fits to the data and predicts the value of *D* under specific conditions.

3.3.1 Modelling the factors that affect the number of decoy matches

3.3.1.1 The effect of precursor length on the mean number of decoy matches

Since post-digest di-peptide precursors can vary greatly in size, we sought to evaluate its impact on false matching and thus compared the average number of decoy matches (*D*) obtained for precursors of different sizes. To begin with, we focused our attention specifically on precursors carrying a single charge, since these are in many cases the most commonly observed precursors. In Figure 3.4, *D* is plotted against the precursor length (*L*), as defined by the number of residues present in the di-peptide precursor, including the sequences of both peptides. The value of *D* increases rapidly as precursor length increases. However, the increase in *D* slows as *D* asymptotically approaches a maximum value. For large precursors the plot flattens to the point of rendering *D* virtually independent of changes in precursor length.

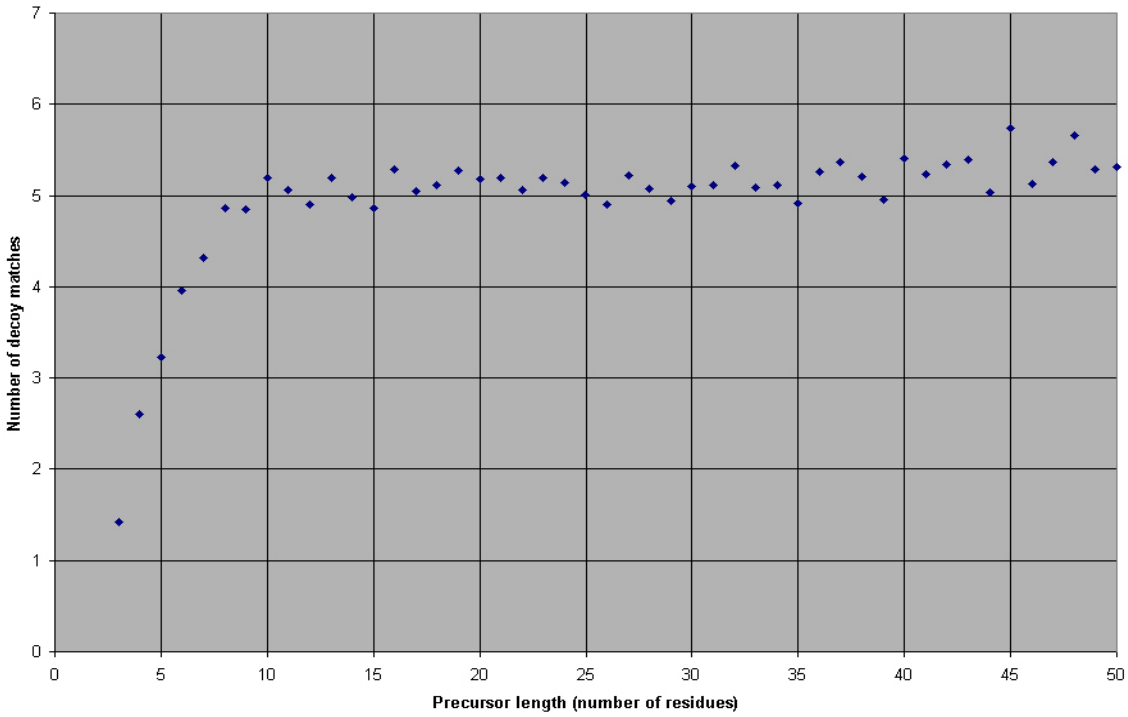


Figure 3.4: The mean number of decoy matches (*D*) versus precursor length (*L*) is plotted for singly charged precursors where fragment peaks within 1ppm of each other were regarded as a match.

3.3.1.2 The relationship between precursor length and theoretical spectrum size

The rapid increase in *D* with respect to precursor length observed for small precursors in Figure 3.4 can be explained by the number of fragment peaks produced by a precursor as a result of collision-induced dissociation (CID) in the mass spectrometer. The more

residues contained in a precursor sequence, the greater the number of inter-residue peptide bonds present, each of which is susceptible to CID. Each peptide bond generates a set of fragments when broken. The greater the number of fragments present in a fragment spectrum, the higher the likelihood that two fragments will by chance have an identical or similar mass, and thus be falsely matched.

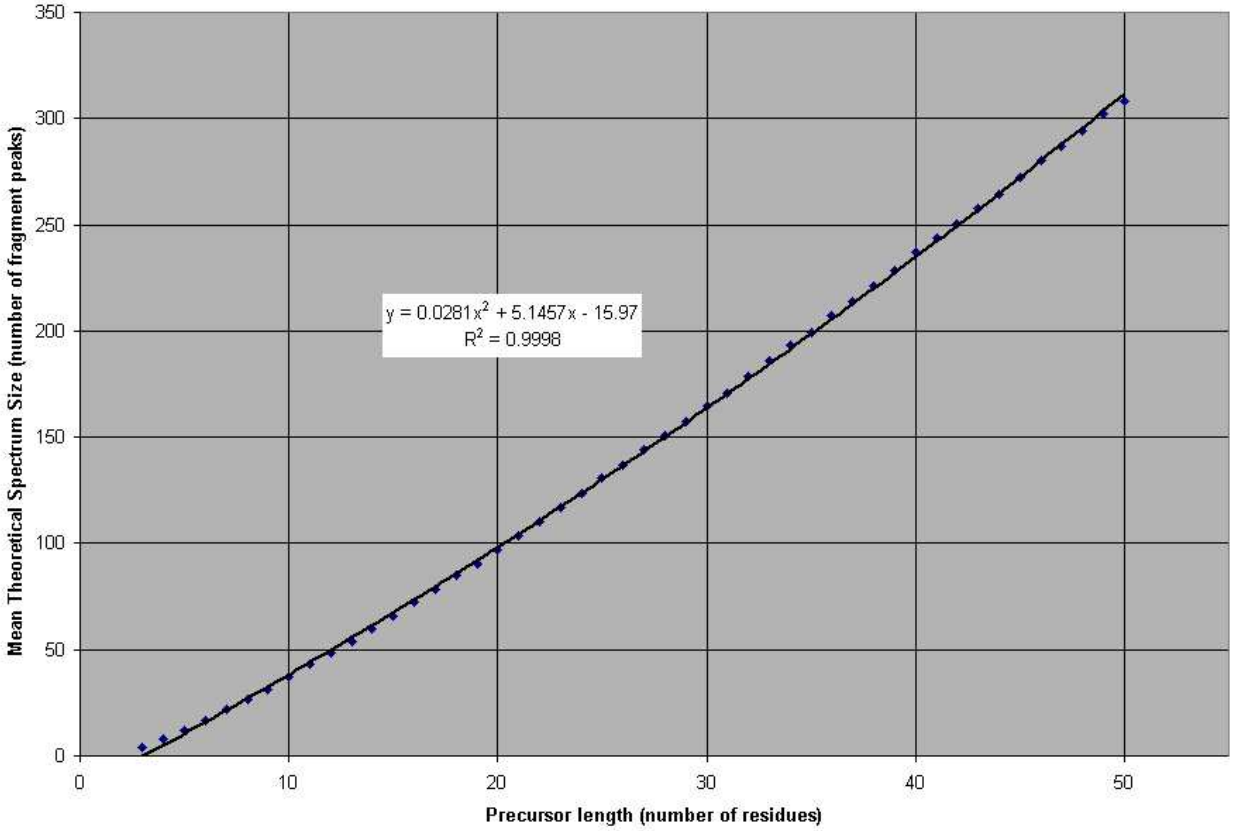


Figure 3.5: *S* is plotted against *L* for singly charged precursors and a second order polynomial is fitted to the data points yielding a Pearson correlation coefficient (R^2) of 0.9998 and a mean error squared of 8.4899.

To illustrate the influence of precursor length on the number of fragments produced, the mean size of the theoretical fragment spectra (*S*), as measured in the number of fragment peaks, was plotted against precursor length in Figure 3.5. It was clear from this graph that *S* increased as precursor length increased. To more precisely describe this relationship, a polynomial curve was fitted to the data points in Figure 3.5 (R^2 = 0.9998; mean of the error squared = 8.4899) revealing that the relationship between *S* and *L* was weakly quadratic (see Equation set 3.1). The plot was close to a linear curve because the fragmentation of each peptide bond consistently produced two fragments. The plot exceeds a linear increase because some amino acid residues are also prone to the possible loss of a neutral chemical group, which creates additional fragments with their own mass (Boersema *et al*., 2009; Hung *et al*., 2007). This factor increases the average number of

fragments produced per precursor residue, which consequently increases the likelihood of two fragments matching by chance.

$$S(L,\ h,\ j,\ k) = h.L^2 + j.L + k$$
$$S(L) = [0.0281].L^2 + [5.1457].L - [15.97]$$

Equation set 3.1: Theoretical spectrum size (*S*) is defined in terms of precursor length (*L*) and the constants *h, j* and *k*.

3.3.1.3 A sequence-dependent mechanism for mass-exact decoy matching

To explain why the rate of increase in *D* with respect to precursor length in Figure 3.5 decreases as precursor length increases, we sought to describe in greater detail the mechanism by which this false or decoy matching occurred. Two peptide fragments that share the same amino acid composition will have the same mass, even if the sequence of their residues differed, and will thus provide an exact decoy match. Figure 3.6 depicts a scenario where a peptide precursor and a sequence-shuffled decoy precursor of equal length can generate fragments with identical masses. For this to occur, the CID fragmentation event must break a particular peptide bond that is positioned within the decoy precursor sequence such that the same collection of residues is present on one side of the broken peptide bond in both the decoy and the true precursor. Not all sequence-shuffled decoy precursors contain a peptide bond whose fragmentation would result in this scenario. In very short peptide precursors this type of scenario is fairly likely, since it is one of only a small number of possible permutations.

For very short precursors, an increase in precursor length increases the number of different mass-identical fragments that are possible for a given decoy precursor. However, as precursor length continues to increase there are an ever expanding number of permutations that do not produce mass-identical fragments and so those permutations that do produce mass-identical fragments occupy an ever smaller portion of the combinatorial space. As the ever slowing increase of decoy matching with respect to precursor length approaches its maximum, the likelihood of additional scenarios that produce mass-identical fragments emerging becomes increasingly unlikely. Near the maximum number of exact decoy matches, only where short peptide fragments are formed, by the fragmentation of a bond close to one of the precursor terminals, are mass-identical fragments sufficiently likely.

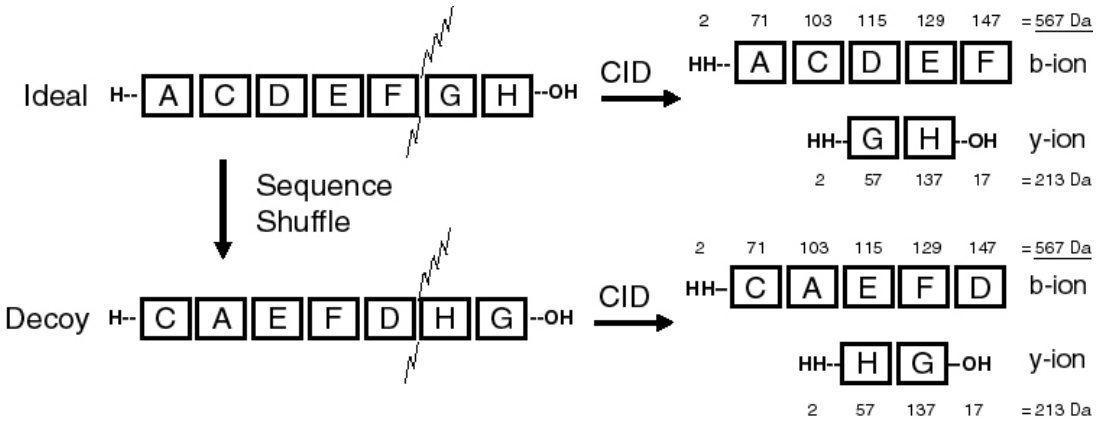


Figure 3.6: Diagram illustrating the occurrence of sequence-based, exact-mass fragment decoy matching. Two sequences differing in residue order may contain the same residue composition, in particular, equivalent fragments of the same ion type, and thus present identical fragment masses for mass spectrometric purposes. The mass in Dalton is displayed immediately above or below each bonded residue and chemical group. For each molecule, its mass is calculated by summing the component masses, displayed in the right margin.

3.3.1.4 Estimating D *from statistical first principles*

As a way of testing the validity of this proposed mechanism, we sought to determine whether the plot shape observed in Figure 3.4 was consistent with the probability values expected as per the proposed decoy matching mechanism, depicted in Figure 3.6. In Equation set 3.2 we show the derivation of an equation for calculating the probability of a mass-exact false match (*M*) occurring in terms of precursor length (*L*) based on this false exact-matching mechanism.

For a decoy match to occur, two chance conditions must coincide: the decoy fragment must be of the same length (event F_{same}) and it must contain the same residues (event R_{same}) as the true fragment. As shown in the first line of Equation set 3.2, the probability of event *M* depends on both these events, and so is calculated as the product of the probability of each occurring.

For the decoy fragment length to be the same as that of the true fragment (event F_{same}), fragmentation must occur at one particular inter-residue peptide bond in the precursor, of which there are as many as the number of precursor residues less one ($L - 1$). The probability of the decoy fragment containing a particular collection of residues given the precursor residue composition (such as event R_{same}), is the fraction of all possible fragment sequences with that residue composition. Because the residue order does not affect the residue composition, the number of fragment sequences with a particular residue composition is calculated as the *F*-combination, where the fragment residues are a sample

from the precursor sequence without replacement (C(*L*, *F*)). Because the order of the residues defines each possible fragment sequence for a fragment of a given length, the number of possible sequences is calculated as the *F*-permutation, where fragment sequence is a sample from the precursor residues without replacement (P(*L*, *F*)). Line 4 of Equation set 3.2 substitutes the factorial form of each. Line 7 of Equation set 3.2 provides a probability function for event *M* purely in terms of precursor length (*L*) by summing the probabilities of all fragment lengths possible for a precursor length, including fragments consisting of only a single residue.

$$1 \quad \Pr(M_F) = \Pr(F_{same}) \times \Pr(R_{same} \mid F_{same})$$

$$2 \quad \Pr(M_F) = \left[\frac{1}{L-1}\right] \times \left[\frac{N_{F=same,\ \wedge\ R=same}}{N_{F=same}}\right]$$

$$3 \quad \Pr(M_F) = \left[\frac{1}{L-1}\right] \times \left[\frac{C_F^L}{P_F^L}\right]$$

$$4 \quad \Pr(M_F) = \left[\frac{1}{L-1}\right] \times \left[\frac{\frac{L!}{F! \times [L-F]!}}{\frac{L!}{[L-F]!}}\right]$$

$$5 \quad \Pr(M_F) = \left[\frac{1}{L-1}\right] \times \left[\frac{1}{F!}\right]$$

$$6 \quad \Pr(M_F) = \left[\frac{1}{[L-1] \times F!}\right]$$

$$7 \quad \Pr(M_L) = \left[\frac{1}{L-1}\right] \times \left[\sum_{F=1}^{L-1} \frac{1}{F!}\right]$$

Equation set 3.2: The probability of a decoy match between fragments with identical mass (*M*) occurring is expressed in terms of the length of the precursor (*L*), the length of the fragment (*F*), the residue composition (*R*) and the number of possible fragments (*N*). The term “same” in the symbol subscript, denotes an identical value for the decoy and true fragment.

The probability of a mass-exact decoy match occurring (event *M*), as calculated by the final line of Equation set 3.2, defines the proportion of fragments in a decoy fragment spectrum likely to be matched to true fragments with the same mass. Therefore, the expected value of *D* can be calculated from *L* by multiplying this probability with the average number of fragments in a decoy spectrum (*S*), as estimated by the final line of Equation set 3.1 (see Equation set 3.3).

$$D(L) = \Pr(M)_L \times S(L)$$
$$D(L) = [\text{Equation } 3.2] \times [\textit{Equation } 3.1]$$
$$D(L) = \left[\frac{1}{L-1}\right] \times \left[\sum_{F=1}^{L-1} \frac{1}{F!}\right] \times [[0.0281].L^2 + [5.1457].L - [15.97]]$$

Equation set 3.3: The mean number of decoy matches (*D*) is expressed in terms of precursor length (*L*).

The equation derived in Equation set 3.3 was used to calculate a set of estimated *D* values which was plotted against precursor length (*L*) in Figure 3.7 to allow a comparison with the simulation data plotted in Figure 3.4. The shape of the two plots was compared to evaluate whether the simulation data shown in Figure 3.4 was consistent with the proposed mass-exact false matching mechanism.

As with Figure 3.4, we observed a rapid increase in *D* for small precursors that gradually slowed as *L* increased in Figure 3.7. However, Figure 3.7 did not display a clear maximum value or prominent horizontal asymptote like that seen in Figure 3.4. If the derived equation of Equation set 3.3 had a maximum value, it was not approached as quickly as the plot in Figure 3.4. The core behaviour of Figure 3.7 was, however, similar enough to that observed in the simulation data to lend credence to, but falls short of fully validating, the proposed mechanism. While Figure 3.7 shared many features with Figure 3.4 the lack of close agreement suggested that the equation derived in Equation set 3.3 may not be suitable for modelling the simulation data plotted in Figure 3.4.

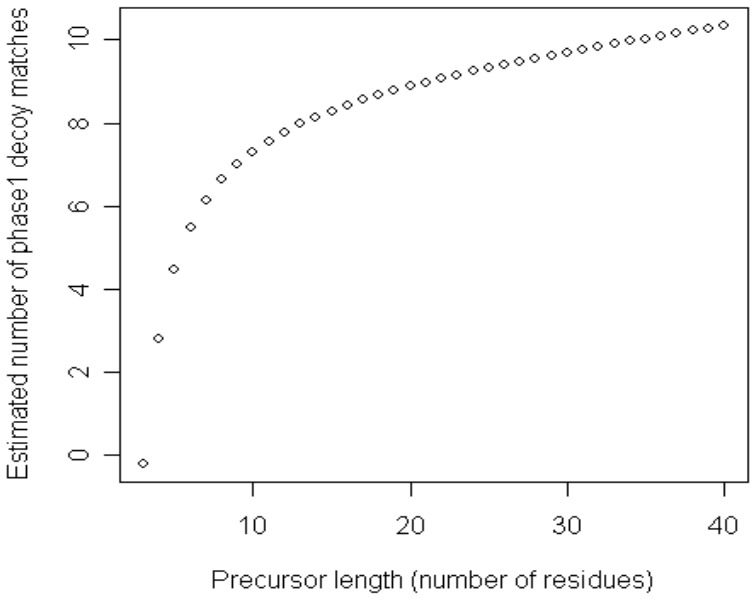


Figure 3.7: The mean number of decoy matches (*D*) for singly charged precursors, as estimated *a priori* from statistical first principles and calculated using the derived equation of equation set 3.3, is plotted against precursor length (*L*).

3.3.1.5 Modelling the relationship between D and L with a calibrated Gompertz function

In an attempt to more accurately predict *D*, a series of mathematical functions were fitted to the data plotted in Figure 3.4 to ascertain if it could be better modelled. Chief amongst these was the Gompertz function, defined in Equation 3.4 (Gompertz, 1825; Laird, 1964). The Gompertz function is essentially a full sigmoid curve modified to shrink the side with positive curvature, effectively turning it into a half-sigmoid curve which increases rapidly and plateaus out towards a horizontal asymptote. The three parameters of the Gompertz function allow precise control over the curvature of the curve making the Gompertz function a versatile modelling tool. In the notation below, the *a* parameter defines the horizontal, maximum-value asymptote, the *b* parameter controls the distance of the centre of curvature from the y-axis and the *c* parameter defines the degree of curvature. Euler's number is denoted by the symbol *e* (≈2.718281828459).

$$Gompertz(L) = a \cdot e^{b \cdot e^{c \cdot L}}$$

Equation 3.4: The Gompertz function (G) defined, in this context, in terms of L as an independent variable.

In Figure 3.8 the Gompertz curve is fitted to the data plotted in Figure 3.4 to determine whether it can appropriately model this data. The calibrated Gompertz function *G()* shown in Equation 3.5 adequately models the data, correlating well with the shape of the plot (R^2 = 0.9427) and fitting the individual data points with a low average error (mean of the error squared = 0.1978).

$$G(L) = [5.23].e^{[-5.4833].e^{[-0.4938].L}}$$

Equation 3.5: The function (G) calibrated against simulation data and expressed in terms of L.

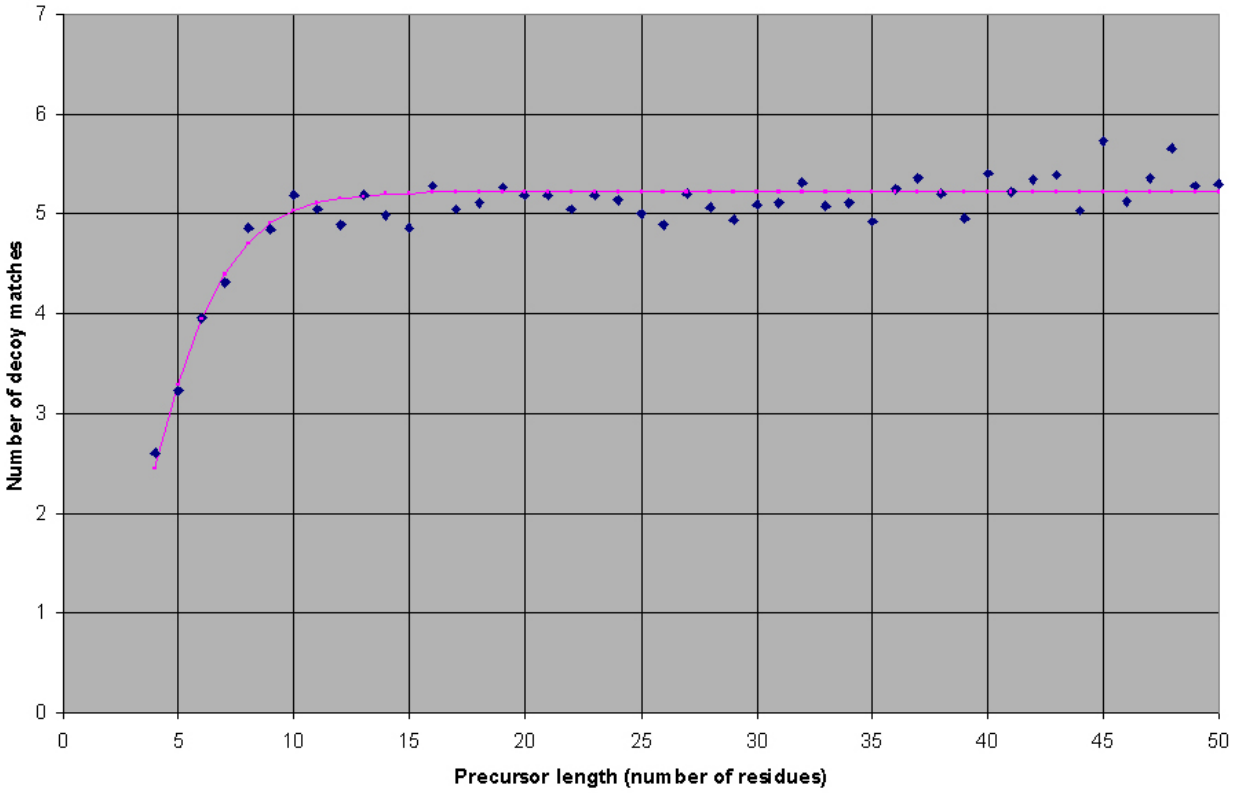


Figure 3.8: The mean number of decoy matches (*D*) for singly charged precursors, where fragment peaks within 1ppm of each other were regarded as a match, is plotted against precursor length (*L*). A Gompertz function is fitted to these data points (see Equation 3.5).

3.3.1.6 Modelling the effect of precursor charge state (Q) on D

Having modelled *D* for singly charged precursors, we sought to investigate the impact of precursor charge state (*Q*) on false matching levels. In mass spectrometry (MS), analyte species are separated within the instrument by means of precise magnetic fields, and are detected on the basis of their charge and mass. Therefore, the analyte molecules must first be ionized before they enter the instrument. Where the instrument uses matrix-assisted laser desorption / ionization (MALDI), a single charge is added to each precursor molecule (Bakhtiar and Nelson, 2000). Where the instrument employs electro-spray ionization (ESI), the analyte precursor can assume any of a number of charge states. The charges carried on each precursor ion persist through the CID fragmentation process, and so are passed on to at least one of the resultant fragment ions, allowing the charged fragment to be detected in the MS^2 spectrum.

In multiply charged precursors, the charges are not always transferred to a single fragment, but may distribute between the fragments, generating two ionized and detectable fragments. The ratio of distributed charges on each fragment may vary greatly,

and is influenced by the ionization properties of each fragment sequence (Paizs and Suhai, 2005). Therefore, the number of charges on the precursor ion (*Q*) defines the maximum number of charges possible for its fragment or product ions. Each possible fragment charge state of a fragment molecule constitutes a unique fragment ion which is observed as a distinct peak in the MS^2 spectrum. Thus, *Q* also defines the number of fragment peaks that can be observed in the MS^2 spectrum for each fragment.

We confirmed this relationship between precursor charge (*Q*) and the number of fragment peaks generated in the theoretical MS^2 spectrum (*S*) by plotting *S* values from the simulation data against *Q* in Figure 3.9. Figure 3.9 showed that *S* was indeed directly proportional to *Q*. This information was used to incorporate *Q* into the equation derived for *S* in Equation set 3.1, shown in Equation 3.6 below.

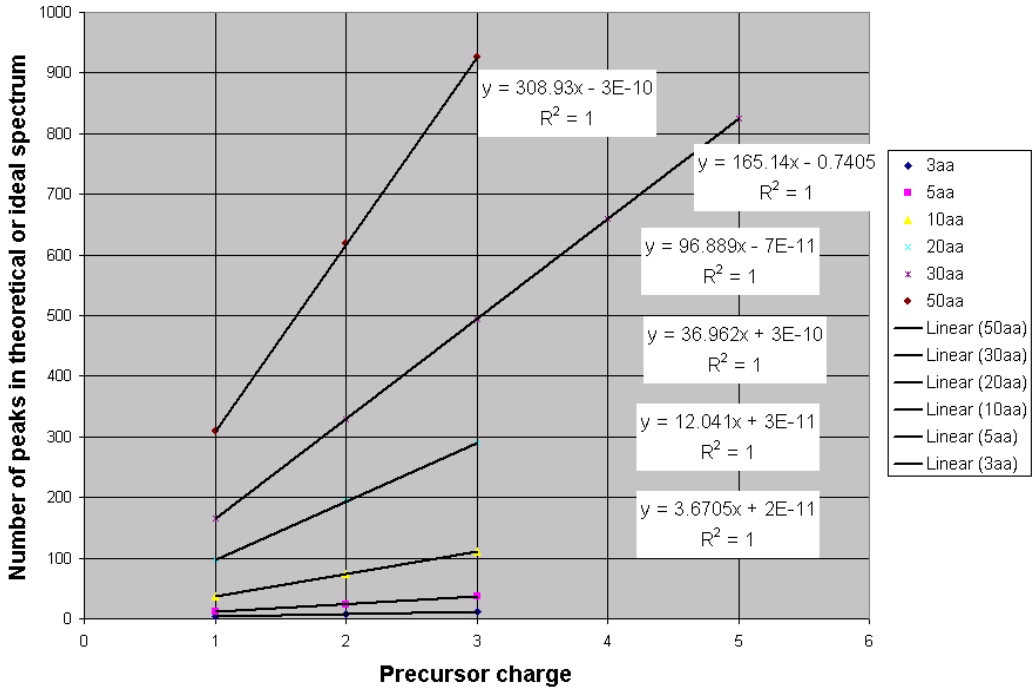


Figure 3.9: The mean theoretical spectrum size (*S*) observed in the decoy matching simulation was plotted against precursor charge (*Q*) for precursors of specified lengths (*L* = (3, 5, 10, 20, 30, 50)) and a linear model was fitted against each set of data points.

$$S(L,\ h,\ j,\ k) = Q.[h.L^2 + j.L + k]$$
$$S(L) = Q.[[0.0281].L^2 + [5.1457].L - [15.97]]$$

Equation 3.6: Mean theoretical fragment spectrum size (*S*) is expressed in terms of precursor length (*L*) and precursor charge (*Q*).

We sought to determine how an increase in *Q* would affect the relationship between *D* and

precursor length (*L*), as previously plotted for singly charged precursors in Figures 3.4 and 3.8. In Figure 3.10, *D* is plotted against *L* for precursors of various charge states. Figure 3.10 showed that an increase in *Q* raised the horizontal asymptote and increased the maximum *D* value which served as an approximation of *D* for very large precursors. To more precisely describe this effect, the estimated maximum value of *D* was plotted against *Q* in Figure 3.11. A linear model was fitted to this data showing that the maximum *D* could be approximated as being directly proportional to *Q*.

The linear increase in maximum *D* value with respect to *Q* arose because the decoy matching of singly charged fragments is mirrored in the fragment set of each additional charge state. If any two singly charged fragments have an identical mass, then those same two fragments will also have identical masses at higher charge states. Because each charge state produces a distinct peak, a single pair of mass-identical fragments resulted in a multiple number of decoy fragment peak matches, proportional to the number of precursor charges (*Q*).

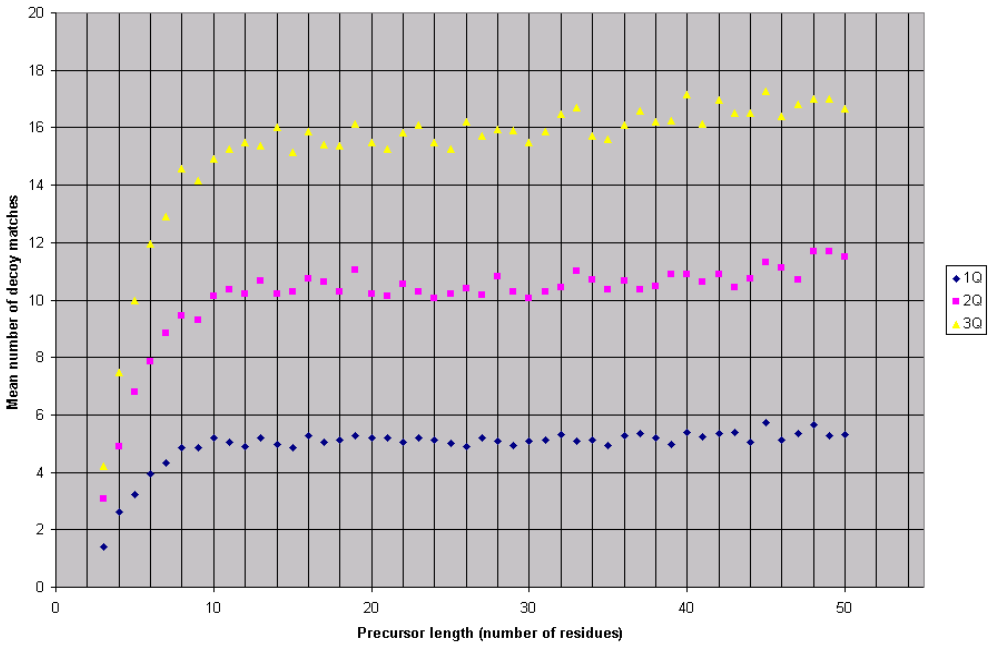


Figure 3.10: The mean number of decoy fragment matches (*D*) is plotted against precursor length *(L)* for precursors with specific precursor charge states (*Q*). *Q* = (1, 2, 3).

Because the maximum *D* value was controlled in the model *G()* by a single parameter, *a* (see Figure 3.8 and Equation sets 3.4 and 3.5), its linear increase with respect to *Q* was easily incorporated into the model, as is shown in Equation 3.7. The constant *a* is substituted by the product of *Q* and a newly created constant, also referred to as *a*. In an attempt to model *D* for multiply charged ions, the derived equation of Equation 3.7 was

fitted to the data (see Figure 3.11) which resulted in Equation 3.8. Figure 3.11 showed that the *G()* function, as modified to consider *Q*, adequately predicted *D* for multiply charged precursors (R^2 = 0.9427; mean of the error squared =1.5451). Equation 3.8 was implemented within the scoring scheme of AnchorMS for estimating the level of exact-mass decoy matching.

$$G(L) = [5.23] \times Q \times e^{[-5.4833] \times e^{[-0.4938] \times L}}$$

Equation 3.7: The mean number of decoy fragment matches (*D*) was estimated by the function *G()*, which was expressed in terms of precursor charge (*Q*) and precursor length (*L*), as well as the constants *a*, *b*, *c* and *e*.

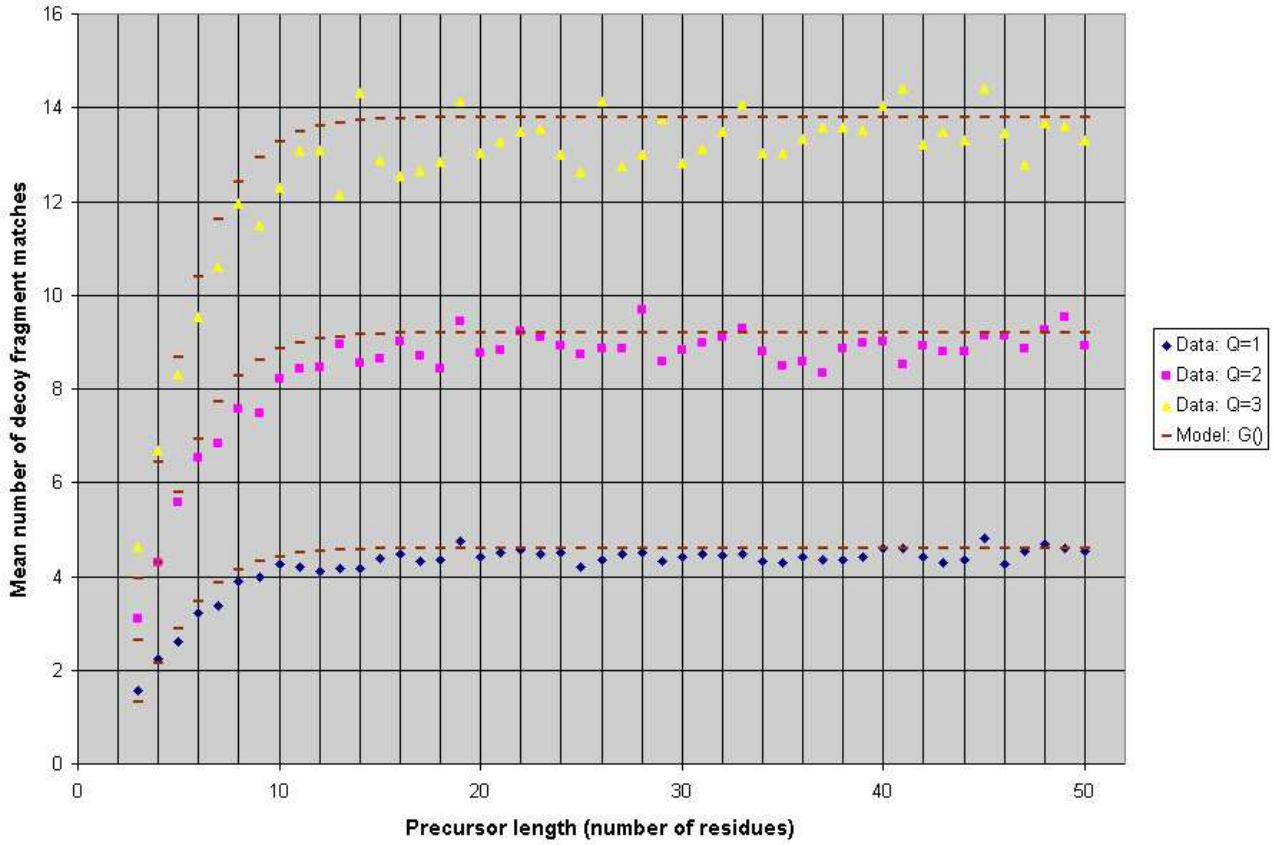


Figure 3.11: The mean number of decoy fragment matches (*D*) was plotted against precursor length (*L*) for specific precursor charge states (*Q* = (1, 2, 3)) and fitted to *G()* as derived in Equation set 3.4.

$$G(L) = [5.23] \times Q \times e^{[-5.4833] \times e^{[-0.4938] \times L}}$$

Equation 3.8: The mean number of decoy fragment matches (*D*) was estimated by the function *G()*, which was expressed in terms of precursor charge (*Q*) and precursor length (*L*).

3.3.1.7 Modelling the effect of tolerance (T) on decoy matching

Thus far we have only considered scenarios where a decoy and true fragment were deemed a match if they had identical masses. This may be termed exact-mass decoy matching. Matching criteria this stringent are not appropriate for MS analyses where imperfect instrument accuracy may result in the apparent mass differing from the true mass of a fragment species. To accommodate peak value variation due to imperfect instrument accuracy, two peaks may be deemed a match if their masses or peak values were within a pre-set distance from each other. This maximum distance which defined a match is termed the tolerance (*T*). The lower the tolerance is set, the greater the number of false negative matches. As tolerance increases, the number of false positives increases because the range of matched values around the expected value of the true fragment increases, and a random, unrelated fragment species was more likely to occur within the larger range of matching values.

Increased decoy matching resulting from an increase in tolerance need not involve exact-mass matches, and so is not dependent on decoy-matched fragments having an identical residue composition. Instead, increased decoy matching resulting from an increased tolerance value arises because a larger tolerance value allows for matching of a greater range of mass or peak values around the true value, and a false peak is more likely to fall into a larger range by chance.

Having successfully derived a model for the number of decoy fragment matches (*D*) in terms of precursor charge (*Q*) and precursor length (*L*), we next tried to incorporate tolerance (*T*) as a parameter. This derived model caters specifically to exact-mass matching, and is only appropriate where tolerance values are very small, requiring virtually exact masses for a match to occur.

To expand the model, we sought to determine how an increase in tolerance (*T*) would affect the relationship between *D* and precursor length (*L*). Figure 3.12 plotted *D* against *L* for a number of tolerance values. At higher tolerance values, *D* became more responsive to increases in *L*. At very low tolerances, the slope of *D* for large precursors with respect to *L* was near linear. However, as tolerance increased, so did the negative curvature of the plot shown in Figure 3.12. This can be explained: the higher the tolerance (relaxed stringency), the greater the chance that another peak will fall within the tolerance of the predicted theoretical peak value. The greater the precursor length, the greater the number of fragment peaks the precursor produces, and the greater the chance that one of them

happens to have a peak value close to a predicted peak value. When the tolerance value is sufficiently low, then *D* becomes insensitive to increases in *L,* because only mass-exact decoy matches are within the tolerance of the predicted peak value, even as the number of decoy fragment peaks increases.

However, for very small precursors in Figure 3.12, the relationship between *D* and *L* was almost unaffected by an increase in tolerance. For these very small precursors, the half sigmoid plot shape observed in Figure 3.10, was maintained, even at high tolerance values. On the left side of Figure 3.12, where a range of short precursor lengths (*L*) was plotted, we saw a region dominated by tolerance-independent decoy matching, as was observed in Figure 3.10 where very low tolerance values were applied. On the right side of Figure 3.12, for larger values of *L*, we saw a region dominated by tolerance-dependent matching. In this second region, where *L* is large, the tolerance-independent matching that dominates the low-*L* region remained approximately constant.

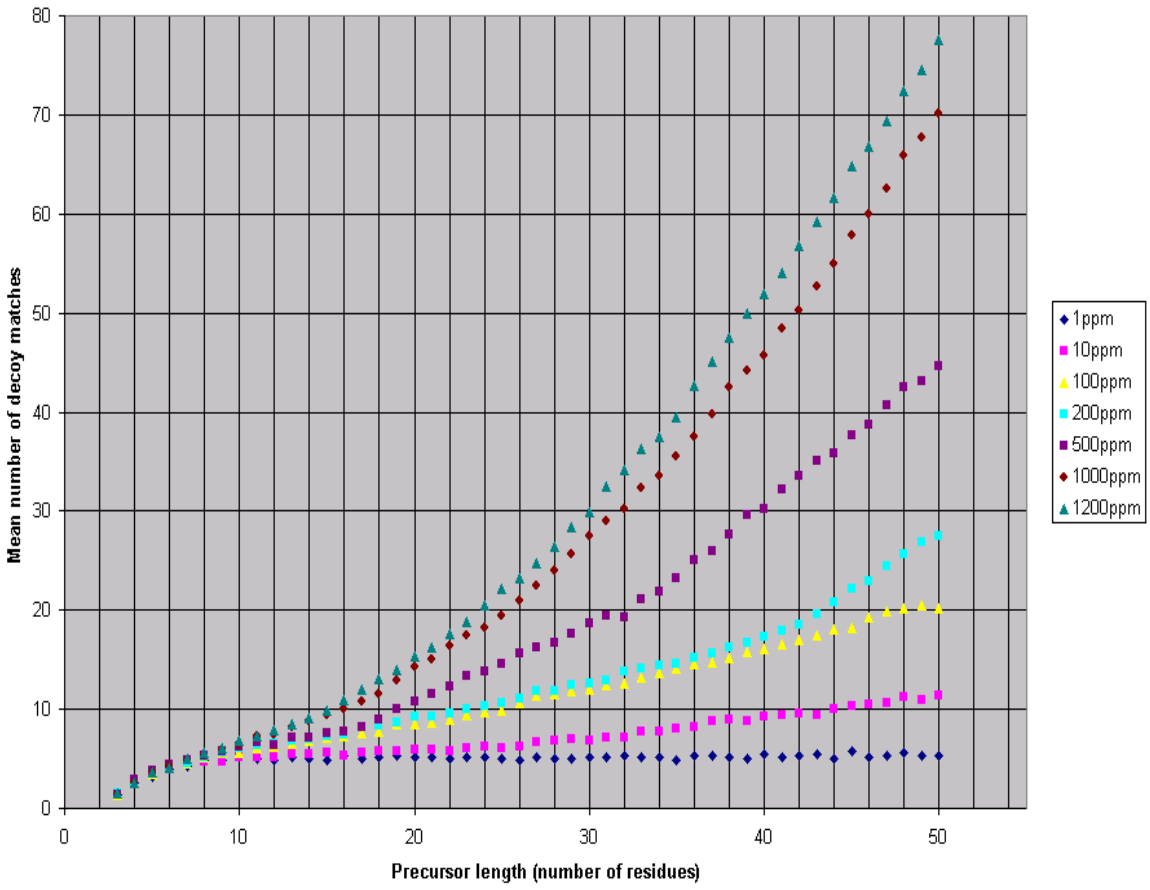


Figure 3.12: The mean number of decoy matches (*D*) was plotted against precursor length (*L*) for singly charged precursors using several tolerance (*T*) values.

Next we considered how an increase in tolerance would affect the relationship between *D*, *L* and *Q*, as depicted earlier in Figure 3.10. In particular, we sought to determine if an increase in tolerance (*T*) would affect the effect that precursor charge had on the relationship between *D* and *L* (as shown in Figure 3.10). In Figure 3.13 we plotted *D* against *L* for singly, doubly and triply charged precursors, applying a higher tolerance (10 ppm) compared to Figure 3.10.

We first considered the left-most region of Figure 3.13: Like Figure 3.10, Figure 3.13 displayed a tolerance independent region for small precursors with a half sigmoid plot shape and, like Figure 3.10, an increase in charge increased the maximum value of this half-sigmoid in *D* with respect to *L*. The increase in *D* with respect to *L* for small precursors (left-most graph region), appears to be dependent on *Q* but largely independent of *T*. This is consistent with exact-mass decoy matching being the dominant form of decoy matching for small precursors. As explained above for Figure 3.10, as *Q* increased, each single-charge, exact-mass also occurred at higher charge states. The observed *T*-independence is because increases in *T* did not increase the likelihood of an exact-mass match. The level of *T*-dependent decoy matching was low amongst small precursors because there were fewer fragments to fall within the tolerance value of a given predicted peak value by chance.

Next we considered the right-most region of Figure 3.13: For larger precursors in the right-most region we observed that, where *D* remained constant in Figure 3.10, in Figure 3.13 *D* became more responsive to *L,* and thus showed a greater-than-linear increase in gradient, as was also seen in Figure 3.12. However, even though the same tolerance (*T*) applied to each of the three plots in Figure 3.13, sharper increases in *D* with respect to *L,* where *L* is large, were observed as precursor charge (*Q*) increased. Therefore, the increase in *D* with respect to *L* for large precursors (right-most graph region), where tolerance is significantly greater than zero, observed in both Figure 3.12 and Figure 3.13, appeared to be dependent on both *Q* and *T*.

There are two reasons why an increased *Q* contributed to increases in *D,* where *T* was significantly greater than zero: Firstly, at higher values of *Q*, as with higher values of *L*, more fragment peaks were produced. Secondly, because the additional peaks were generated with higher precursor charge states (*Q*), increases in Q resulted in an increased peak density. Both these factors increased the likelihood of a fragment peak falling within the tolerance of the peak value of a predicted peak value by chance.

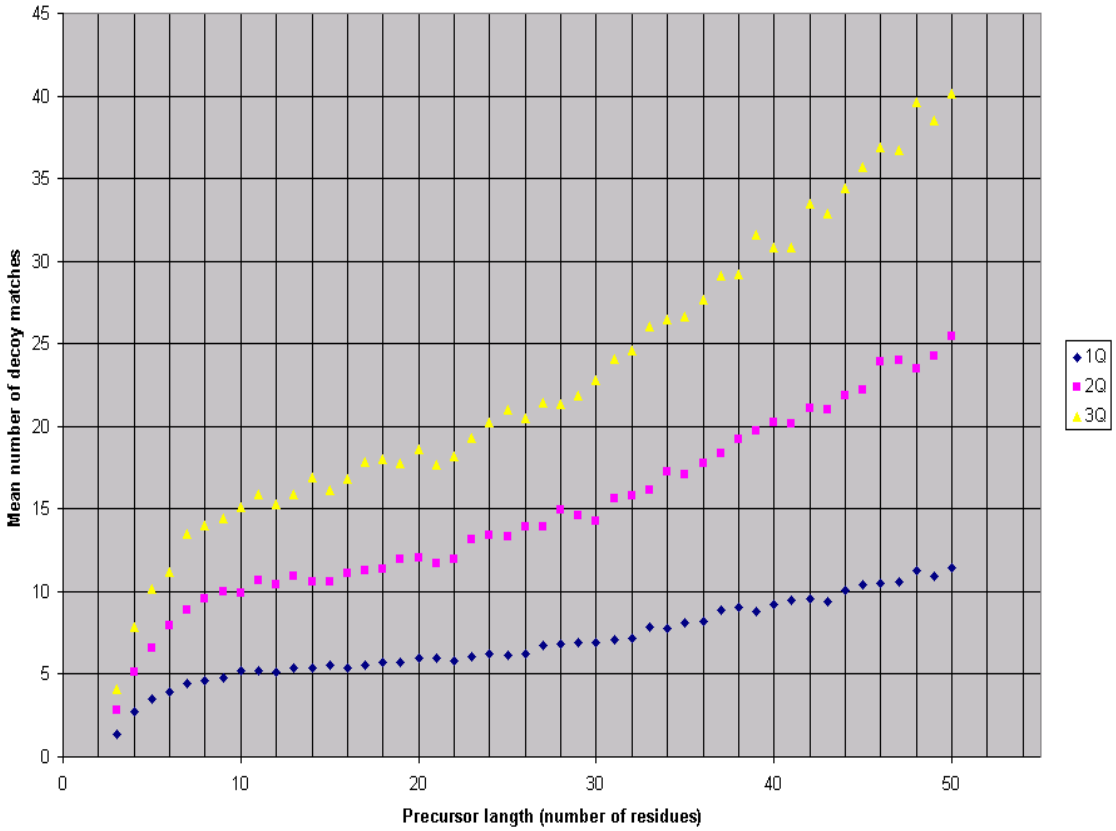


Figure 3.13: The mean number of decoy matches (D), where a tolerance of 10 ppm was applied, was plotted against precursor length (L) for several precursor charge states (Q = (1, 2, 3)).

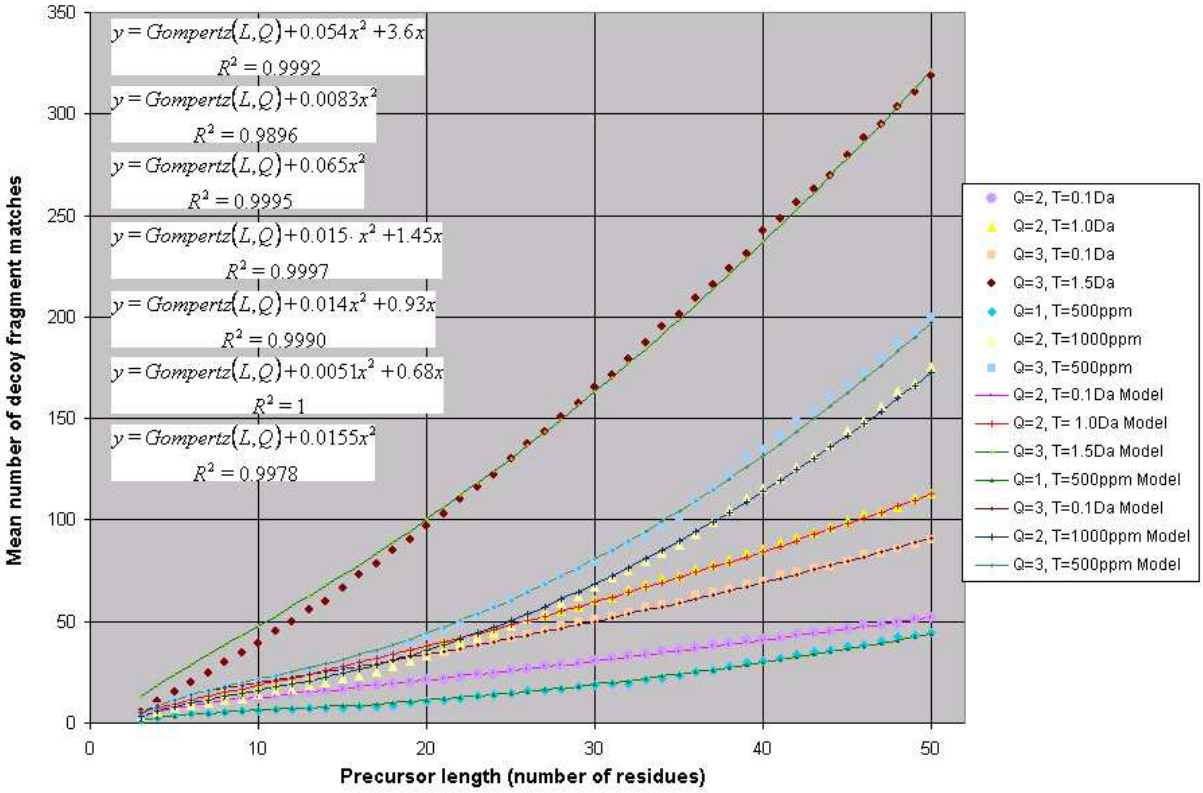
Earlier in this Chapter the exact-mass decoy matching which was observed in the left-most region of Figures 3.12 and 3.13, was described and modelled. According to this model, for large precursors, the value of *D* for exact-mass decoy matching approached a maximum. Tolerance-dependent matching increased *D* beyond this maximum in the right-most region of Figures 3.12 and 3.13, though the exact nature of this increase remained unclear. Because exact-mass and tolerance-dependent decoy matching occur concurrently, we propose that *D* can be modelled as the sum of exact-mass and tolerance-dependent decoy matching.

We next addressed two related ideas. Firstly, we asked whether *D* could, in fact, be modelled as the sum of mass-exact and tolerance-dependent decoy matching. Secondly, we set out to find the best model with which to describe the tolerance-dependent increase in *D*. Consequently, we fitted a number of mathematical functions to the plot observed in the right-most region of Figures 3.12 and 3.13. Of these, a second degree polynomial was selected as the best fit to the data. This model was added to the exact-mass matching model, *G()*, to form a putative general model for *D*. In Figure 3.14, below, we fitted the sum of *G()* (modelling exact-mass matching) and a polynomial model to a selection of

plots for *D versus L*, each with a different *Q* and *T* value.

This fitted model provided a good description of the plotted data (mean R^2 = 0.9938). Figure 3.14 suggested that the tolerance-dependent increase in *D* is well approximated as quadratic, and that *D* can indeed be modelled as the sum of *G()* and a second degree polynomial. This general model for *D* is shown in Equation set 3.9, below.

A



B

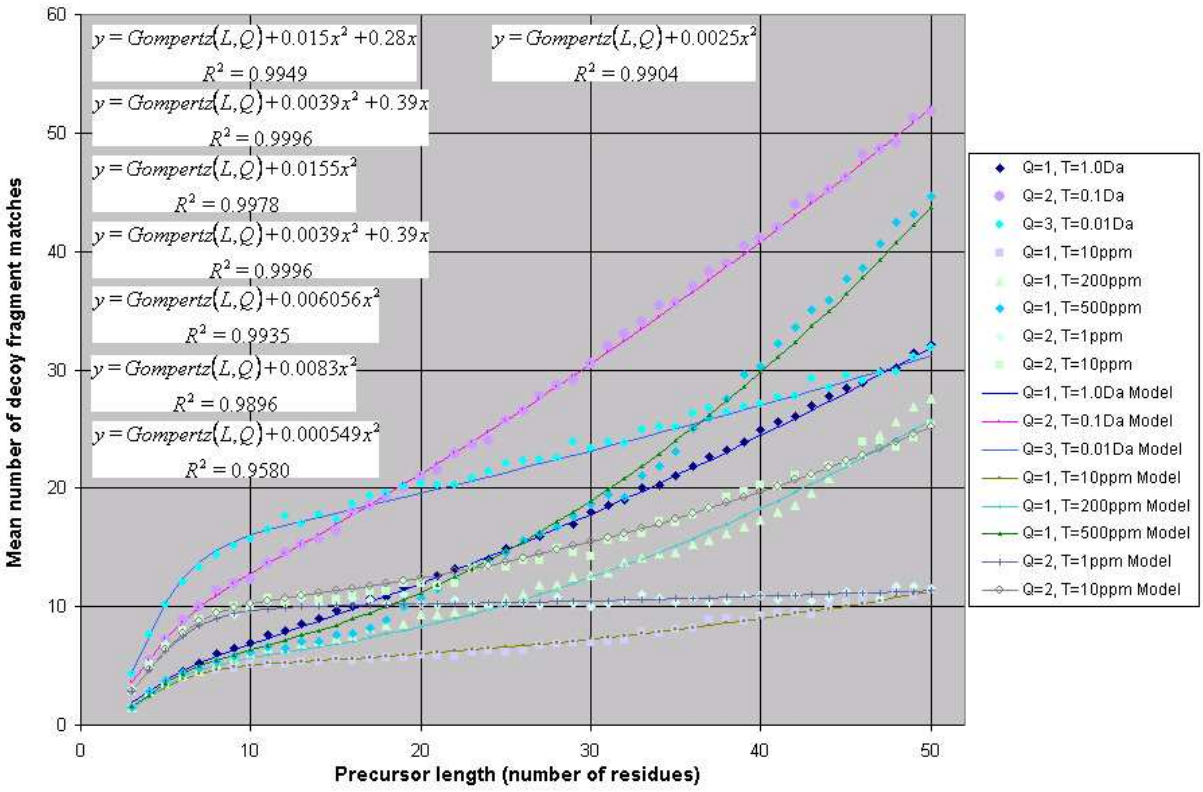


Figure 3.14: The mean number of decoy fragment matches (*D*) was plotted against precursor length (*L*) for various values of *Q* and *T*, and the sum of *G()* and a second degree polynomial were fitted to each plot. For the sake of visual clarity the plots are divided between panels (A) and (B).

Alongside the data plots, Figure 3.14 also showed the fitted equations of the second degree polynomial component in the model for each of the data plots. We wondered how the polynomial component of the fitted models varied with regard to changes in *Q* and *T,* since it was evident from the equations shown alongside Figure 3.14 that the specific polynomial in each of the fitted models differed from one another. In particular, the coefficients of L^2 and *L* (represented by *f* and *d*, respectively, in Equation set 3.9) varied as the value of *Q* or *T* changed. Both *f* and *d* increased in response to an increase in either *Q* or *T*. In cognisance of this, the coefficients *f* and *d* were each represented in Equation set 3.9 as functions (*f(Q,T)* and *d(Q, T))* which were, themselves, dependent on precursor charge (*Q*) and tolerance (*T*).

$$D = D_T + D_E$$
$$D = \left[f.L^2 + d.L\right] + G(L,Q)$$
$$D = f(Q,T).L^2 + d(Q,T).L + a.Q.e^{b \times e^{c \times L}}$$

Equation set 3.9: The mean number of decoy fragment matches (*D*) was expressed in terms of precursor length (*L*), and in terms of the coefficient functions *f(Q, T) and d(Q,T).*

In Figure 3.14, some of the *T* values applied were absolute tolerance values, where the tolerance distance (*T*) was measured in Daltons. Other *T* values in Figure 3.12 were relative matching tolerance values (in ppm), where the tolerance distance (*T*) was defined as a fraction of the total peak value of the observed peak. We compared the fitted equations of the polynomial components shown alongside Figure 3.14 to discern whether the use of absolute (Da) or relative (ppm) matching tolerance affected either *f* or *d*, as represented in Equation set 3.9. Indeed, where *T* was measured as relative tolerance, the value of *d* in the fitted polynomial tended to be very close to zero in comparison to the value of *f*. This constituted a significant difference between the data plots matched with absolute and with relative tolerance. In the case of absolute tolerance (Da), both *f* and *d* increased as *Q* and *T* increased. In the case of relative tolerance (ppm), *d* could be approximated to a constant value of zero, and was therefore independent of both *Q* and *T*.

In order to adequately model *D* in terms of precursor charge (*Q*) and tolerance (*T*), we attempted to describe the functions in Equation set 3.9, namely *f(Q, T)* and *d(Q, T)*. In pursuing this objective, we chose to separately consider absolute (Da) and relative (ppm) tolerance values. For each, we began with the general model in Equation set 3.9, but derived a different model in each case to describe *D* in terms of *L*, *Q* and *T*. The modelling of *D* for absolute and for relative tolerance is described in two separate sections, below.

3.3.2 Modelling D in terms of precursor charge (Q) and tolerance (T), measured as absolute tolerance

Equation set 3.9 showed a general model for the mean number of decoy fragment matches (*D*). Here, *D* was modelled as the sum of exact-mass (D_E) and tolerance-dependent (D_T) decoy matching ($D_E + D_T$). The component representing the mean number of decoy fragment matches due to tolerance-dependent decoy matching (D_T), was expressed as a second degree polynomial in terms of precursor length (*L*), and in terms of the two functions *f(Q, T)* and *d(Q, T)*. We sought to expand this component (the model for D_T) so as to provide a better description of D_T in terms of only *L*, *Q* and *T*, where *T* was measured as absolute tolerance ($D_{T, Da}$).

3.3.2.1 Modelling $D_{T,Da}$ *in terms of the mean theoretical spectrum size (S)*

The general equation used to describe $D_{T,Da}$ was a second degree polynomial (see Equation set 3.9), expressed in terms of precursor length (*L*). Earlier in this chapter, the mean theoretical spectrum size or number of expected fragment peaks (*S*) was also modelled using a second degree polynomial expressed in terms of precursor length (*L*), as shown in Equation set 3.1. Given this parallel, and given that both *D* and *S* can be expressed in terms of *L*, we speculated that $D_{T,Da}$ could be modelled in terms of *S*.

To investigate this possibility, the mean number of decoy fragment matches (D_{Da}) was plotted against mean theoretical spectrum size (*S*), as shown in Figure 3.15, below. In Figure 3.15 the data plots appeared to be linear for all but very short precursors. In particular, the first four data points, which correspond to precursor lengths of 3 to 6 amino acid residues, deviated from the linear shape of the remaining data points. Figure 3.15 also showed the fitting of a linear model to the data points representing precursors larger than 6 residues for each precursor charge state (*Q*). The good fit achieved for precursors larger than 6 residues (mean R^2 = 0.9988; mean of the error squared = 0.060), confirms the linear nature of the data for this range of *L* values. However, this linear model provided a very poor approximation for *D* where precursors were smaller than 7 residues. Nonetheless, we could see in Figure 3.12 that tolerance-dependent decoy matching makes a minimal contribution to the overall number of decoy fragment matches observed for very small precursors (L < 6). Therefore, we accepted the linear model as a good approximation for the number of decoy fragment matches due to tolerance-dependent decoy matching ($D_{T,Da}$). The linear relationship between $D_{T,Da}$ and *S* was expected because each additional peak in the theoretical spectrum increased the probability of a

decoy match by a fixed amount. Viewed differently, the probability of a decoy match with a particular observed peak that considered the entire theoretical spectrum, was equal to the sum of the probability to match for each theoretical peak individually. Therefore, $D_{T,Da}$ was equal to the probability of a single peak match multiplied by the number peaks, which is a linear function in terms of the number of theoretical fragment peaks (S).

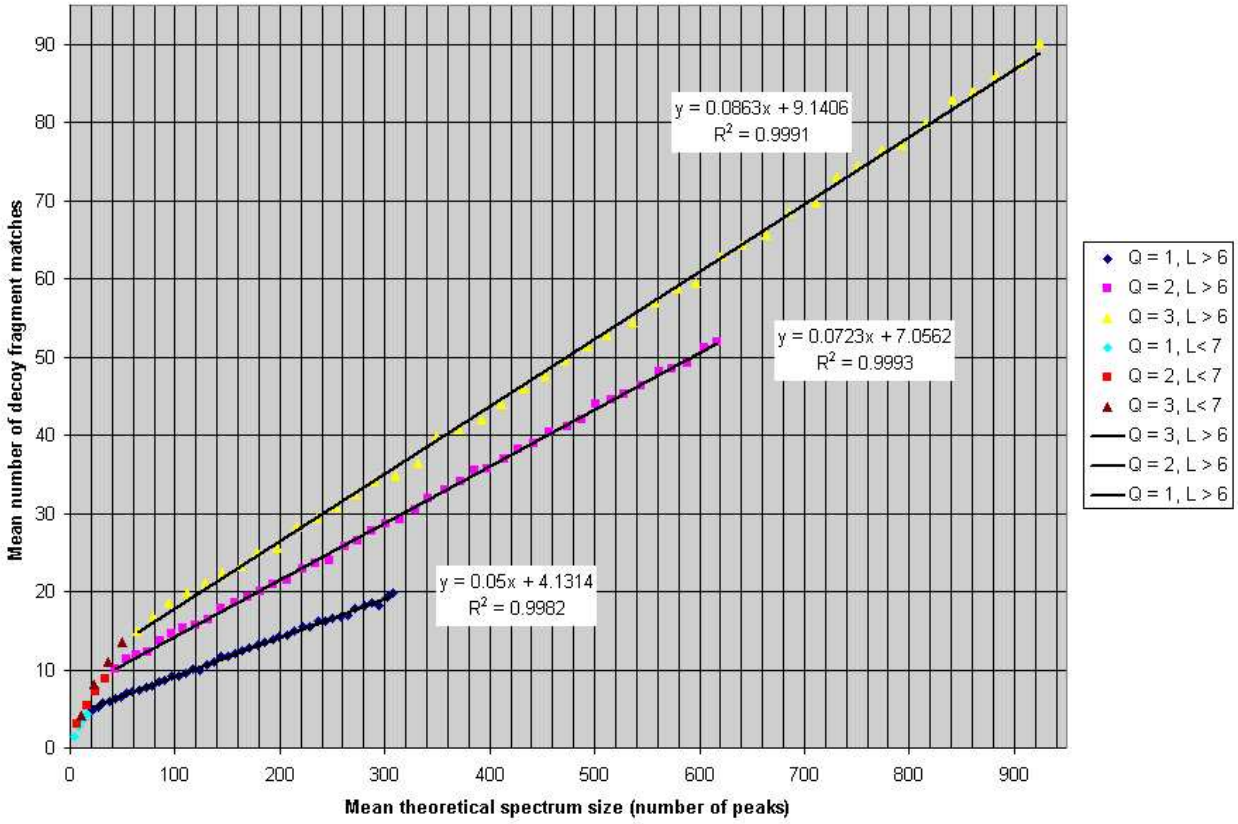


Figure 3.15: The mean number of decoy fragment matches (D_{Da}), where an absolute tolerance of 0.1Da was applied during matching, was plotted against the mean theoretical spectrum size (S), for precursor lengths (L) 3-50 residues and various precursor charge states (Q = (1, 2, 3)).

In equation set 3.10, $D_{T,Da}$ was expressed as directly proportional to S, based on the assumption that the linear model was a fair approximation. The slope for $D_{T,Da}$ *versus* S, was defined as the parameter P. The parameter P could be expressed as $D_{T,Da}$ divided by S, and so was equal to the ratio of $D_{T,Da}$ to S. P could also be interpreted as the mean fraction of theoretical fragment peaks that were decoy matched. Therefore, P represented a meaningful measure that could be related directly to decoy matching of peaks within mass spectra.

Having defined P, we wondered whether P was a constant or a function dependent on other variables, such as L, Q or T. In Figure 3.15, L varied between data points and

increased (moving from left to right across the plot) because *S* increased as *L* increased (see Equation 3.1 and Figure 3.5). Despite this change in *L,* the approximated linear gradient of each data plot in Figure 3.15 remained constant for precursors larger than 6 residues. Figure 3.15, as well as the fitted equations alongside it, showed that the linear gradient (*P*) varied with precursor charge (*Q*). Therefore, *P* was independent of *L* but dependent on *Q*. To confirm whether or not *P* was dependent on *T*, we referred back to Figure 3.12 which showed that *D* varied at a given precursor length (*L*) and precursor state (*Q*), depending on the tolerance value used during matching, due to tolerance-dependent decoy matching (D_T). D_T was, therefore, *T* dependent. If one of the equations in Equation set 3.10 is rearranged to isolate *P* on one side of the equation, as in the last line of Equation set 3.11, we can then see that *P* is dependent on $D_T(L, Q, T)$ and is therefore, itself, a *T*-dependent function. In order to describe $D_{T,Da}$ in terms of *Q* and *T*, we next attempted to model *P(Q, T)*.

$$D_{T,Da} = P \times S(L,Q)$$
$$D_{T,Da} = P \times \left[Q \times \left[h \cdot L^2 + j \cdot L + K\right]\right]$$
$$D_{T,Da} = P \cdot Q \cdot \left[\left[0.0281\right] L^2 + \left[5.1457\right] L + \left[-15.97\right]\right]$$

Equation set 3.10: The mean number of decoy fragment matches due to tolerance-dependent decoy matching with an absolute tolerance applied ($D_{T,Da}$) was expressed in terms of mean theoretical spectrum size (*S*) and the parameter *P*. In line 3, *S* is substituted by the final model for *S (L, Q)* as defined in Equation 3.6.

3.3.2.2 The relationship between P *and tolerance (*T*)*

To better understand how the value of *P()* was affected by changes in tolerance (*T*), we plotted *P()* against tolerance (*T*), as shown in Figure 3.16, below. For the sake of clarity, we first considered only singly charged precursors. Figure 3.16 showed that *P()* increased as tolerance increased. This was expected since an increase in tolerance would theoretically increase the likelihood of a theoretical peak value falling within the tolerance of the predicted peak value by chance, resulting in a decoy match. However, *P()* did not undergo a uniform increase with respect to *T* across the *T* value range. In Figure 3.16 we observed a series of narrow regions wherein *P()* underwent very rapid increases with respect to *T*. These rapid-increase regions specifically occurred at intervals of 1 Da along the x-axis. Between these rapid-increase regions *P()* was less sensitive to changes in tolerance (*T*). Overall, the presence of alternating rapid and negligible *P()* increases along the range of *T* values created a step-like plot shape.

$$D = D_T + D_E$$
$$D = [P \times S] + G$$
$$P = \frac{D - G}{S}$$
$$P(Q,T) = \frac{D(L,Q,T) - G(L,Q)}{S(L,Q)}$$

Equation set 3.11: An equation for *P()* in terms of *D*, *G()* and *S* was derived.

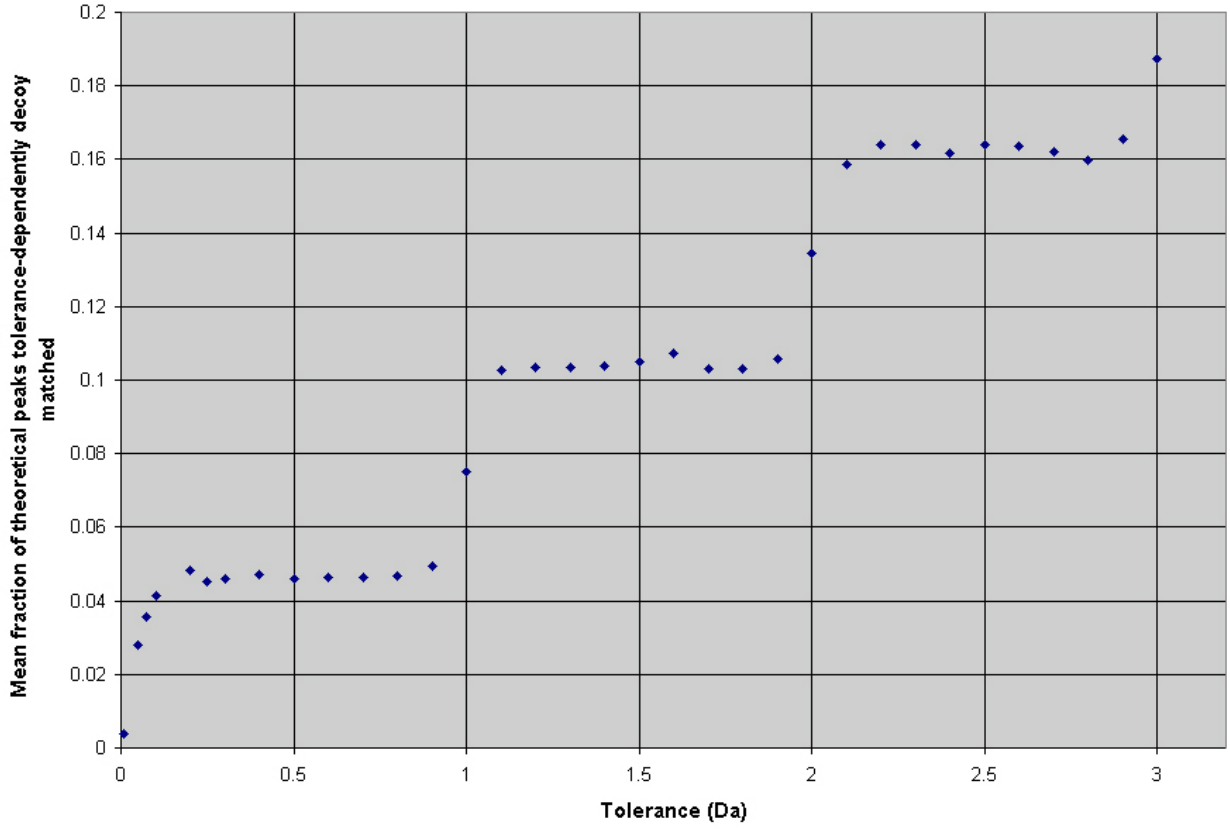


Figure 3.16: The mean fraction of theoretical fragment peaks decoy matched (*P*) was plotted against tolerance (*T*) in Daltons (Da), for singly charged precursors.

The 1 Dalton intervals at which *P* increased in Figure 3.16 could be explained in terms of statistical differences in peptide mass. Based on the observation of 1 Da interval increases in Figure 3.16, we wondered if certain differences between the masses of randomly generated peptides were more prevalent within the combinatorial space of peptide species that could be compared. Two peptide sequences were randomly generated and the difference in their predicted masses observed. Figure 3.17 summarized these results and indicated that peptide mass differences also tended to occur at around 1 Dalton intervals. The higher incidence of peptide mass differences close to 1 Dalton intervals was likely due to the fact that atoms will differ by a multiple of approximately 1 Dalton, since this is the approximate mass of one proton.

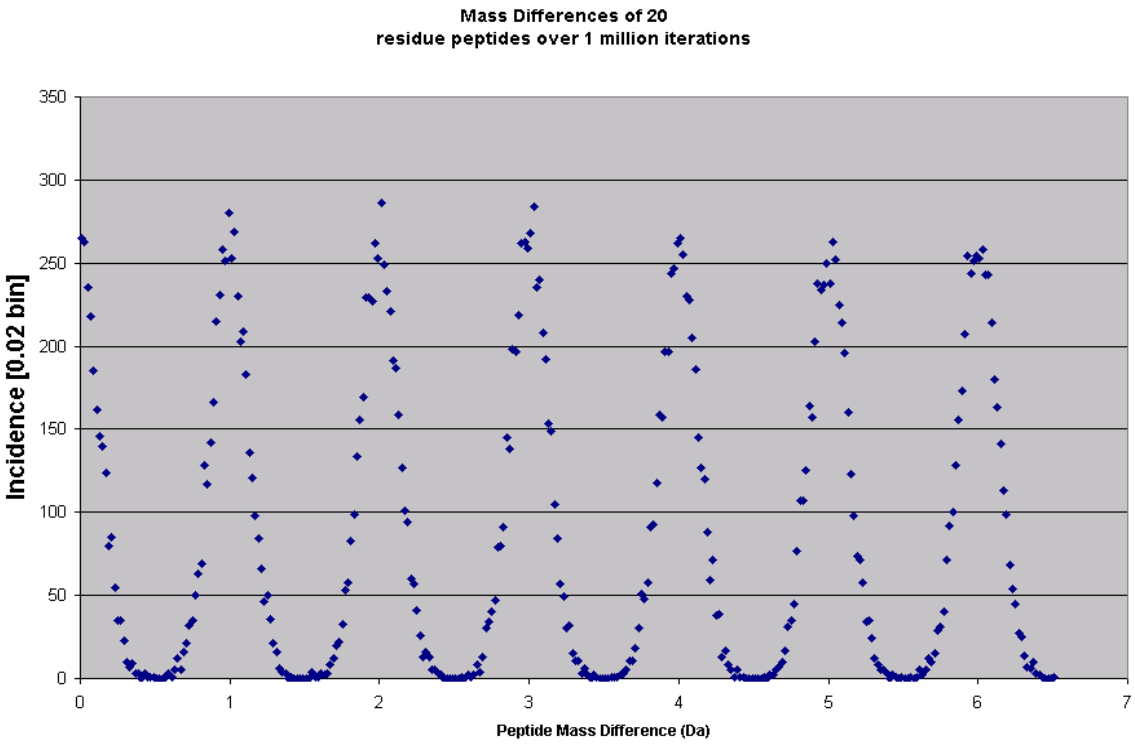


Figure 3.17: The number of observations within each bin was plotted against peptide mass difference values, where each bin groups the observations over a 0.02Da range of mass difference values. Peptide sequences of up to 20 residues were generated and compared, and this was repeated 1 million times.

In order to better understand, and eventually model, the data plot in Figure 3.16, we examined several features of the plot in greater detail. We wondered whether the value of *P* increased by the same amount in each rapid-increase zone observed in Figure 3.16, and if this rate of change in *P* could be described in terms of *T*. The simplest way to measure the change in *P* across one interval was to note the difference in the *P* values observed at the mid-points of each interval. For this purpose, the mid-point of the rapid-increase zone or the mid-point of the negligible-increase zone in Figure 3.16 could be used since both lie on the mid-point of the overall plot. We chose to incorporate the maximum number of mid-point data points. In Figure 3.18 below, the *P* values at the mid-point of each rapid-increase zone and each negligible-increase zone were plotted against tolerance (*T*).

The plot in Figure 3.18 showed the overall shape of the plot in Figure 3.16, without the fluctuations that form the “steps”. That is, it indicated the path of the steps. If a set of mid-points formed a line, then the rate of change was constant, and *P* changed by the same amount across each interval. However, if the mid-points did not form a line, then the rate of change of *P* was not constant and changed with tolerance.

In Figure 3.18 we could see that the *P* values at the interval mid-points formed a line. A linear model was fitted to the data, and correlated well with the data points (R^2 = 0.9983; mean of the error squared = 0.07920655). This suggested that, on average and across multiple tolerance intervals, an increase in tolerance (*T*) proportionally (linearly) increased the likelihood of a peak falling within the tolerance of a predicted peak, resulting in a decoy match.

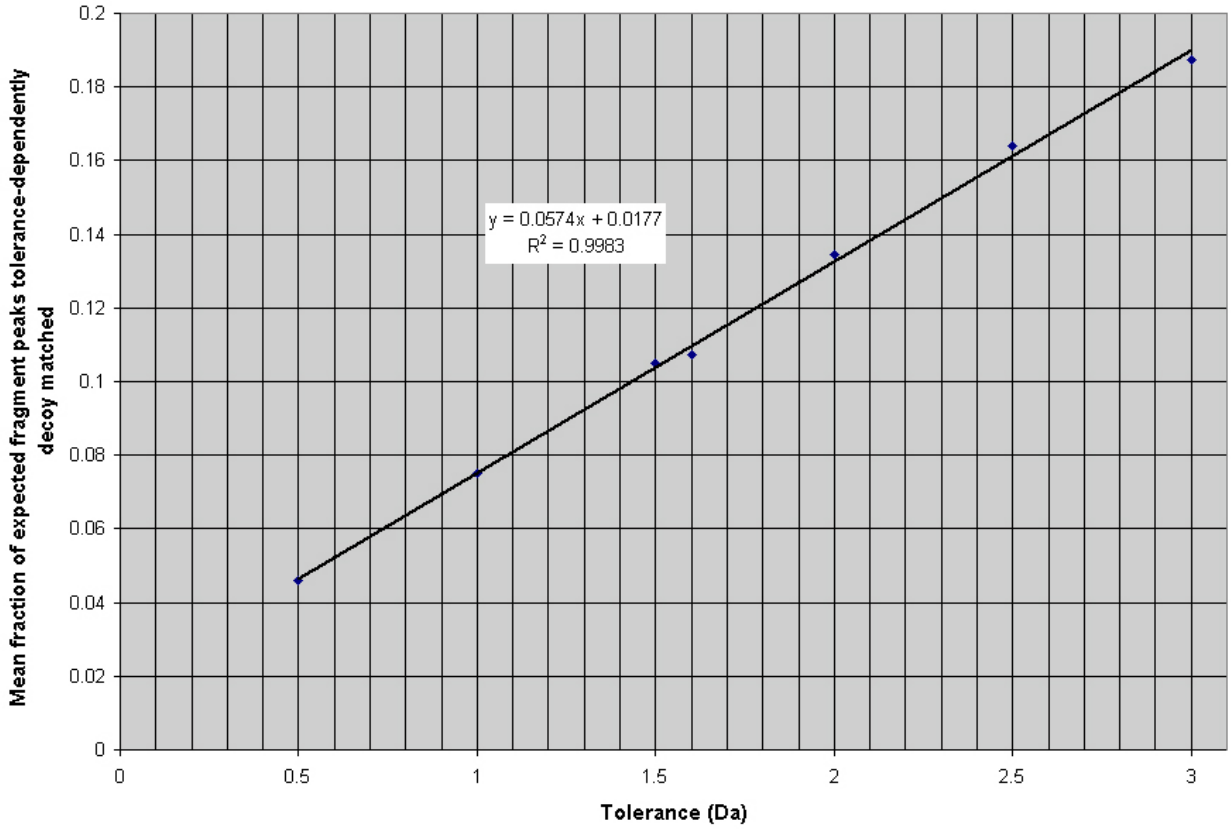


Figure 3.18: The fraction of theoretical peaks tolerance-dependently decoy matched (*P*) at the mid-points of the rapid-increase and negligible-increase zones in Figure 3.16 were plotted against the tolerance (*T* = (0.5, 1, 1.5, 2, 2.5, 3)) values in Daltons at those mid-points.

3.3.2.3 The influence of precursor charge (Q) on the relationship between P and tolerance (T)

3.3.2.3.1 An increase in Q changes the relationship between P and T

We next sought to determine how a change in precursor charge (*Q*) would affect the relationship between *P* and tolerance (*T*). In Figure 3.19 below, the *P* value was plotted against tolerance (*T*) for a number of precursor charge states (*Q*). We noted that an increase in precursor charge (*Q*) significantly altered the relationship between *P* and *T*. Figure 3.19 suggested that precursor charge (*Q*) affected both the intervals between rapid-

increase zones, as well as the overall shape of the plot along the mid-points. Let us consider each of these effects in turn.

3.3.2.3.2 An increase in Q reduces inter-step intervals

As precursor charge (*Q*) increased, the intervals between zones of rapid increase in *P* shorten proportionately. For example, rapid-increase zones occurred at intervals of 1 Da for singly charged precursors, but at intervals of 0.5 Da for doubly charged precursors, which is the interval for singly charged precursors divided by two. However, the amount by which *P* increased across each rapid-increase zone decreases as precursor charge (*Q*) grew. These two changes resulted in the apparent dissipation of the "steps" in the plot with increasing *Q* as their relative contribution to the total value of *P* declined. For precursor charge states (*Q*) of 4 and larger, the plot started to resemble a smooth curve and the presence of steps in the plot was no longer visible.

3.3.2.3.3 Reduced intervals are due to division by Q when calculating peak values

The proportional shortening of the interval between rapid-increase zones as *Q* increased could be explained in terms of how the peak value was calculated from the mass of a fragment. Equation 3.12 shows how the calculation of the peak value of a fragment ion involves dividing the total mass (*M*) by the charge it carries (*z*). By this calculation, the distance between singly charged peaks would be reduced as *Q* (the maximum fragment *z*) increased. For example, any two decoy-matched, singly charged peaks that were approximately 1 Da apart, would only be approximately 0.5 Da apart as doubly charged ions.

$$Peak\ \ value = \frac{M + \left[z \times M_{hydrogen}\right]}{z} = \frac{M + \left[Q \times M_{hydrogen}\right]}{Q}$$

Equation 3.12: Peak value is calculated using analyte mass (*M*), the mass of a hydrogen ion ($M_{hydrogen}$) and the ion charge (*z*). The precursor charge (*Q*) defines the maximum potential ion charge (*z*).

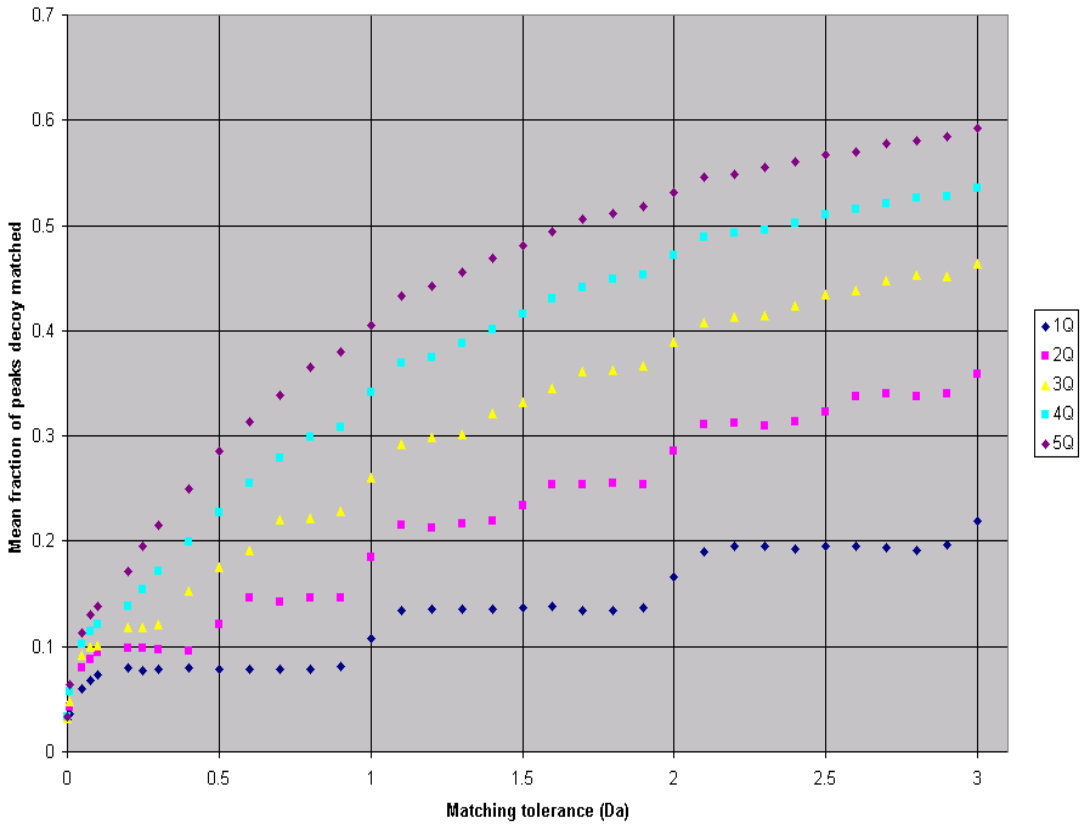


Figure 3.19: The fraction of theoretical peaks tolerance-dependently decoy matched (*P*) was plotted against tolerance (*T*) in Daltons for precursors carrying between 1 and 5 charges (*Q* = (1, 2, 3, 4, 5)).

3.3.2.3.4 Increased Q reduces linearity of plot shape through the interval mid-points

The second effect of increased *Q,* as seen in Figure 3.19, was an apparent change in the overall plot shape. As *Q* increased, the plot shape appeared to become increasingly sigmoidal, with a rate of change that declined as tolerance (*T*) increased. To more precisely test this possibility, the *P* values at the plot mid-points of each plot in Figure 3.19 were plotted against tolerance (*T*) in Figure 3.20. To gauge the linearity of mid-points for each precursor charge state (*Q*), line equations were fitted to each mid-point plot in Figure 3.20. For singly charged precursors the plot was shown to be perfectly linear. But, as *Q* increased, the mid-points became increasingly non-linear with lower rates of change for large tolerance (*T*) values.

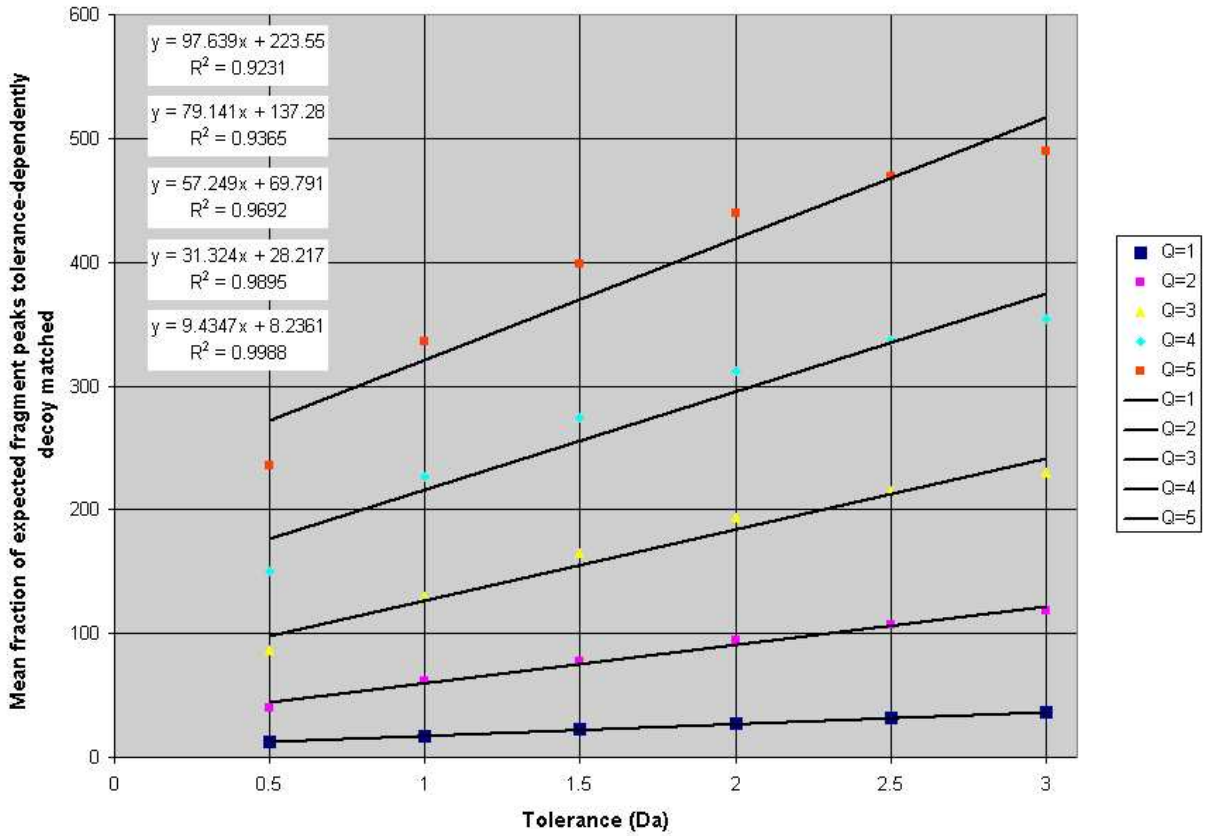


Figure 3.20: The fraction of theoretical peaks tolerance-dependently decoy matched (*P*) at the mid-points of the rapid-increase zones in Figure 3.19 were plotted against the tolerance values in Daltons at those mid-points (*T* = (0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5)), for multiple precursor charge states (*Q* = (1, 2, 3, 4, 5)). A line equation is fitted to each plot.

To better describe the effect of precursor charge (*Q*) on the relationship between *P* and *T*, we attempted to model the mid-point values of *P*, as plotted in Figure 3.20, in terms of both tolerance (*T*) and precursor charge (*Q*).

3.3.2.4 Modelling the mid-point curve

Preliminary interpolation and manual inspection of the data suggested that the mid-point plots in Figure 3.20 may be consistent with a power model, as defined in the first line of Equation set 3.13. To test this possibility a power equation was fitted to each of the mid-point data plots, as shown in Figure 3.21. Figure 3.21 indicated that a power function provided a reasonable approximation of the data (mean $R^2 \approx 0.99102$; mean of the error squared = 1.39×10^{-4}). The applied power function is shown in Equation set 3.13 below, and contained two parameters, namely the base (*CEF*) and exponent (*PWR*) coefficients. Here, the essentially linear trend observable for small charge values was treated as a special case of the power function where the exponential term was approximated to equal 1.

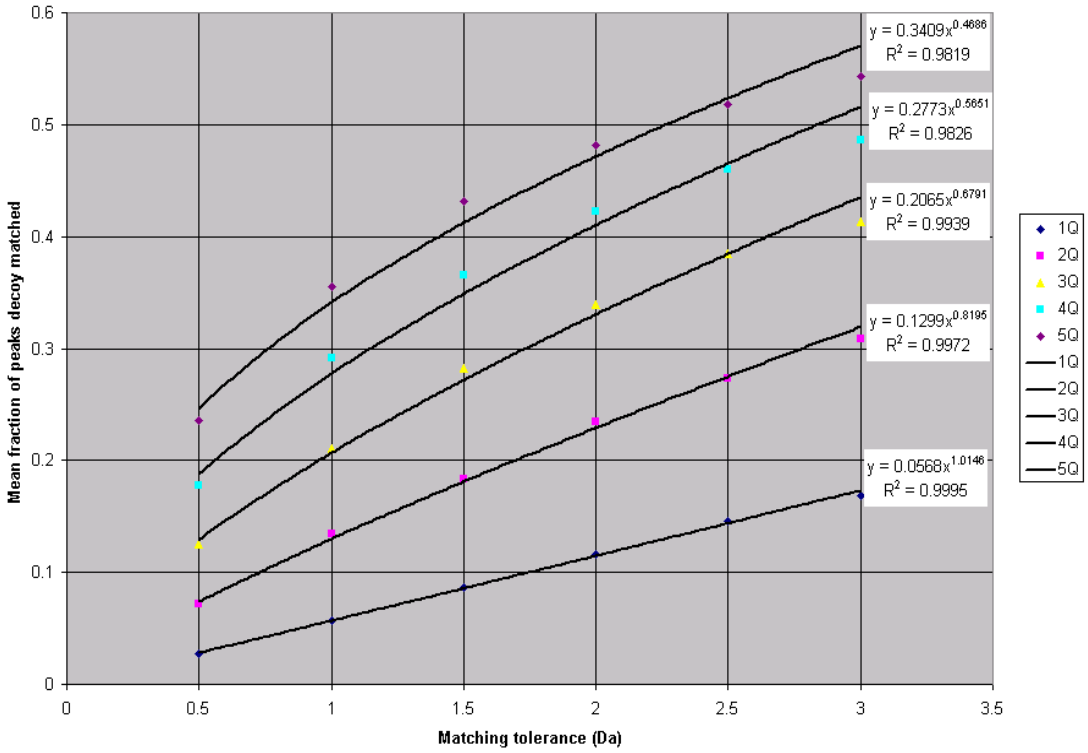
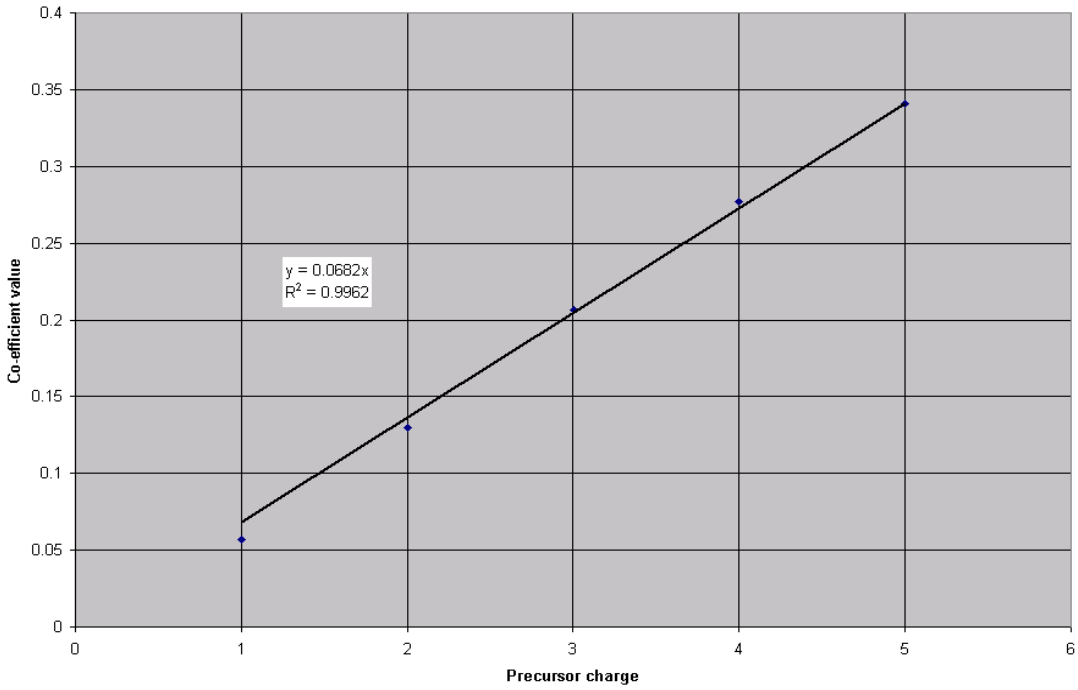


Figure 3.21: The mean fraction of theoretical peaks tolerance-dependently decoy matched (*P*) at 0.5 Da intervals (mid-points) was plotted against tolerance (*T*) in Daltons for multiple precursor charge states (*Q*) and a power model was fitted to each plot.

The calibrated *CEF* and *PWR* parameter values from the fitted power function varied in value between plots (see Figure 3.21), indicating that both *CEF* and *PWR* were dependent on precursor charge (*Q*). In order to model mid-point *P* in terms of *Q*, we attempted to model the power function parameters *CEF* and *PWR* in terms of precursor charge (*Q*). The calibrated values of *CEF* and *PWR* were plotted against Q in panels (A) and (B) of Figure 3.22, respectively.

Several functions were fitted to each of these data plots, but a fitted linear equation was found to be a reasonable approximation for both (mean $R^2 \approx 0.9881$; mean of the error squared = 3.911×10^{-4}). In particular, the base coefficient value (*CEF*) was directly proportional to *Q*. In both these linear models the predicted values were not precise for singly charged precursors, but became increasingly accurate as *Q* increased.

A



B

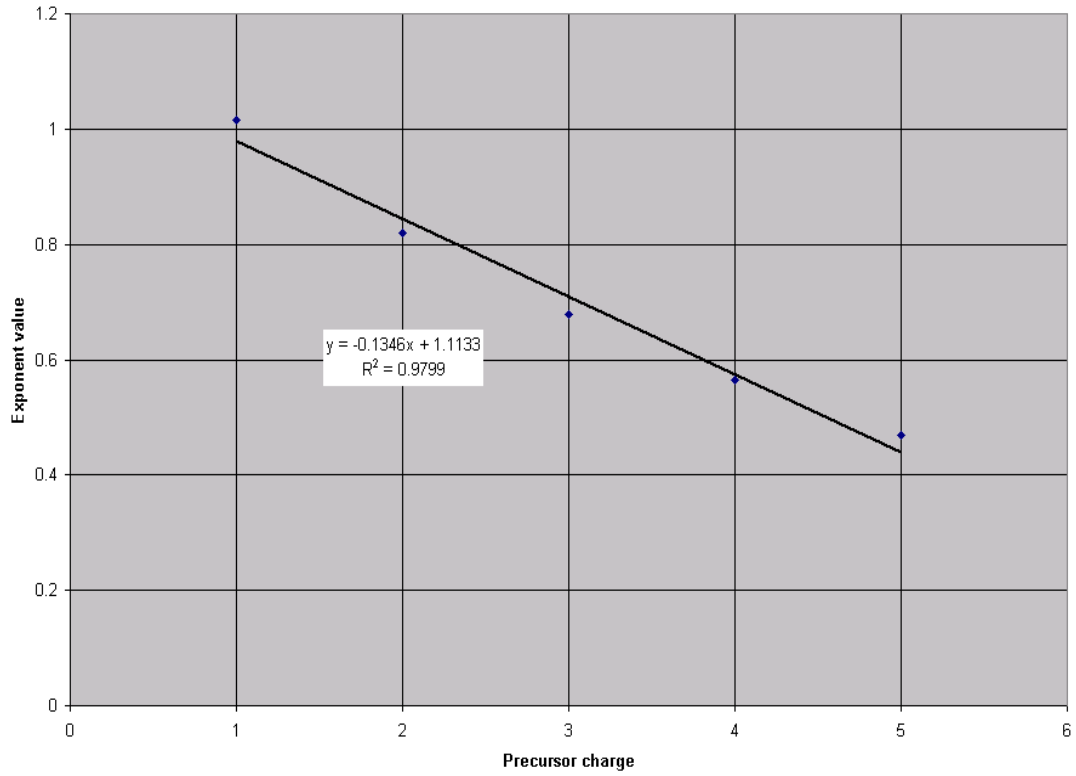


Figure 3.22: Calibrated parameter values within the data-fitted power model for the fraction of expected peaks tolerance-dependently decoy matched (*P*) were plotted against precursor charge (*Q*). A linear equation was fitted to each of the plots. Panels A and B feature the base (*CEF*) and exponent (*PWR*) coefficient parameter values, respectively.

$$P_{Da,\ midpointshape} = CEF \times T^{PWR}$$
$$P_{Da,\ midpointshape} = [cef \times Q] \times T^{pwr0+[pwr1 \times Q]}$$
$$P_{Da,\ midpointshape} = [0.0682 \times Q] \times T^{1.1133+[0.1346 \times Q]}$$

Equation set 3.13: The mean fraction of theoretical peaks tolerance-dependently decoy matched (P) for the mid-point plot shape is expressed in terms of tolerance (T) and precursor charge (Q). In the final line, the calibrated values are substituted into the equation.

Both linear models were incorporated into the power function in Equation set 3.13, above. In so doing, the last line of Equation set 3.13 defines the mid-point values for P in terms of precursor charge (Q) and tolerance (T). To gauge the efficacy of this model, the last line of Equation set 3.13 was plotted alongside the mid-point values of P against T for multiple values of Q in Figure 3.23. Figure 3.23 showed that this model ($P_{Da,\ shape}()$) provided a good description of the mid-point P data ($R^2 \approx 0.9940$; mean of the error squared = 1.39×10^{-4}).

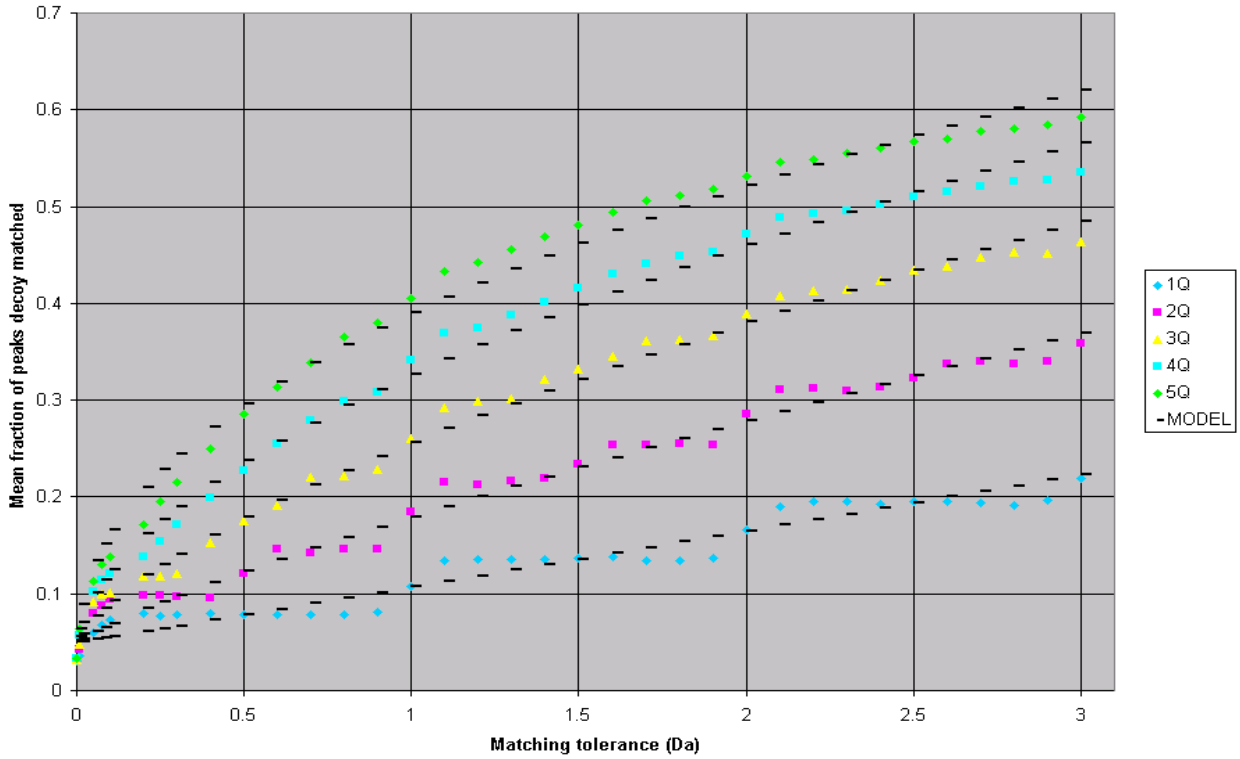


Figure 3.23: The calibrated model ($P_{Da,\ shape}()$) for the mean fraction of theoretical peaks tolerance-dependently decoy matched (P) along the plot mid-points of Figure 3.19 (0.5 Da intervals), was plotted against tolerance in Daltons.

3.3.2.4.1 *P* is modelled as the sum of mid-point shape and periodic deviation

Modelling the mid-point values of *P* has allowed the description of shape and trajectory of the plot in Figure 3.19 with respect to tolerance (*T*) and precursor charge (*Q*). Having derived an equation to describe the mid-point values of *P*, we wondered if this equation could be employed to better model the entire plot of *P* values shown in Figure 3.19. We considered the possibility that *P* could be modelled in terms of *T* and *Q*, as the sum of the mid-point *P* curve derived in Equation set 3.13 and the periodic deviation from this mid-point curve. We thus attempted to model this periodic deviation from the plot mid-point curve, proceeding with the working assumption that it would be possible to model *P* in this way. To begin with, we considered only singly charged precursors. Following that, we incorporated the influence of higher *Q* values.

3.3.2.5 Modelling the periodic deviation of P *from the mid-point of the data plot*

3.3.2.5.1 Modelling the periodic deviation of P from the mid-point of the data plot for singly charged precursors

In order to generate a putative model for the periodic deviation in *P* from the mid-point curve ($PerDev_{Da}$), we subtracted the *P* values predicted by the mid-curve, from observed *P* values plotted in Figure 3.19. These values for singly charged precursors were plotted in blue against tolerance (*T*) in Figure 3.24. As can be seen in Figure 3.24, the data plot was periodic with respect to increasing tolerance (*T*). Due to its apparent similarity to trigonometric plots, we considered the option of modelling this data with a trigonometric function. Also in Figure 3.24, a sine function, adjusted for amplitude and for phase (period = 1 Da), was fitted to the data (see Equation 3.1). With the two plots superimposed in Figure 3.24, we can see that the periodic deviation ($PerDev_{Da}$) is “skewed” towards a vertical axis at intervals of 1 Dalton, in contrast to the “unskewed” trigonometric defined in Equation 3.1.

$$y = [0.02] \cdot \sin(2\pi \cdot T)$$

Equation 3.14: A sine function is expressed in terms of tolerance (T) in Daltons, and is transformed to match a 1 Da periodicity.

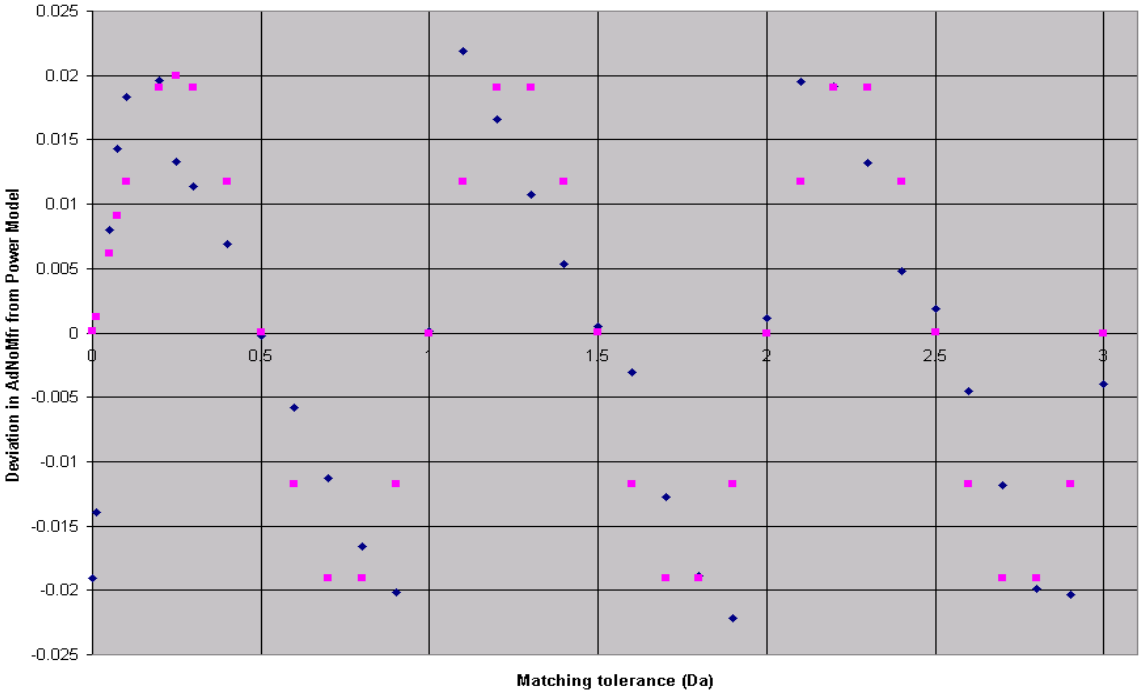


Figure 3.24: The periodic deviation between observed *P* values and those predicted by the calibrated mid-point curve (last line of Equation set 3.13) was plotted against tolerance (*T*) for singly charged precursors. A phase-adjusted (phase = 2π) sine function (Equation 3.1) was fitted to the data.

3.3.2.5.2 Derivation of the custom trigonometric function skewsine()

In order to model this unconventional plot shape, we devised a function to represent the "skewed sine" form (summarized below as Equation set 3.15). This function was called s*kewsine()* and normalized the input angle, θ, according to its relative position within the periodic cycle of the sine waveform (see Figure 3.25 below). As shown in Figure 3.25, the s*kewsine()* function was "skewed" such that its absolute value was symmetrical around vertical axes at periodic "vortex" values (*V*). The input value of parameter θ is converted into a distance from *V* value (D_V).

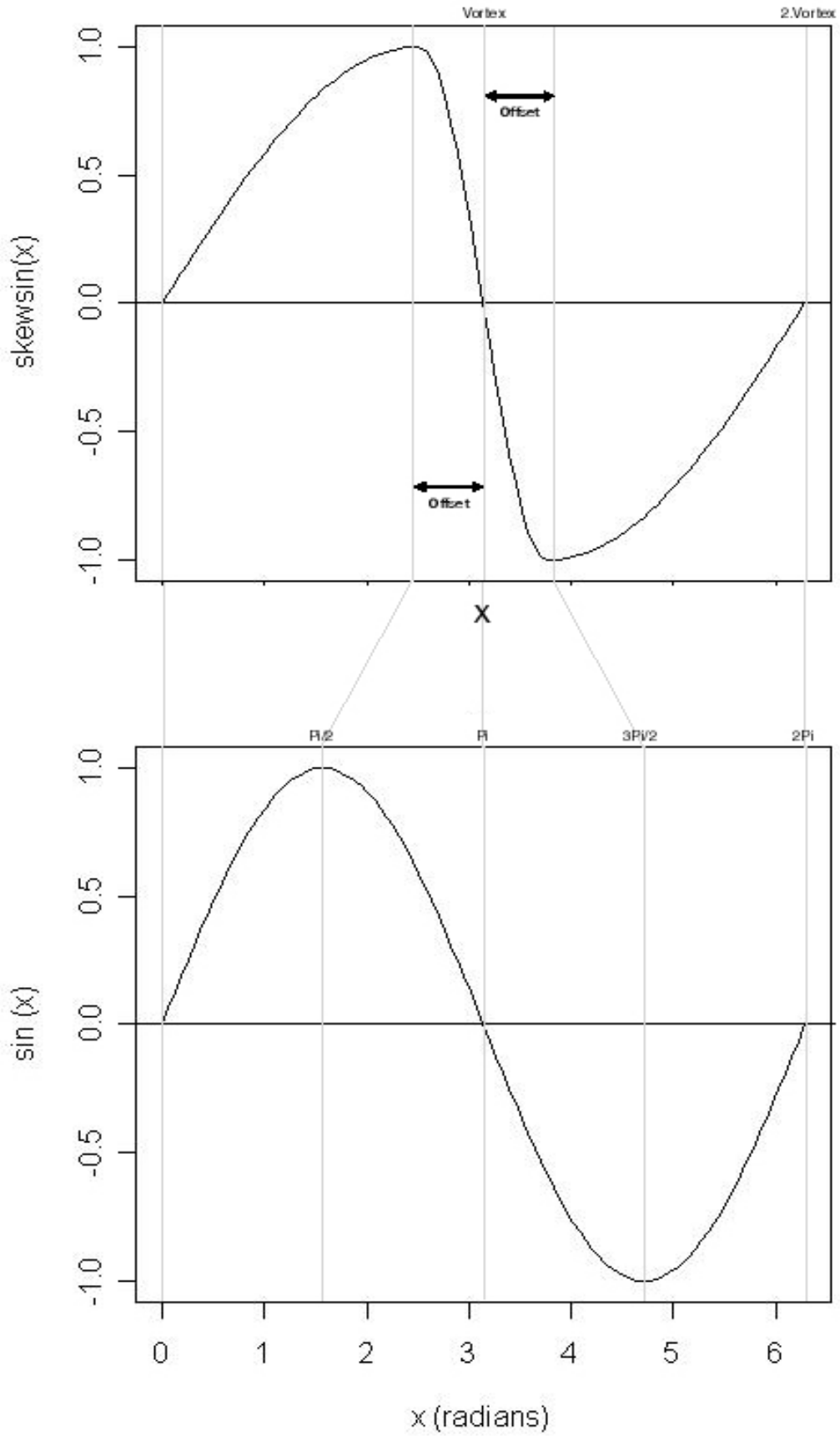


Figure 3.25: Diagram showing the rationale and parameters within the function *skewsine()*. *V* or "Vortex" refers to the angle around which the waveform is distorted. *Sh* or *Shift* refers to $\pi/2$ less the displacement of the apex or apogee (*offset*) of the s*kewsine()* graph from that of *sine()*. D_v refers to the distance between the angle θ and the "vortex" value *V*, or its nearest multiple.

$$\theta_{framed} = frame(\theta) = \text{modulus}(|\theta|, 2\times V)$$

$$D_v = |V - (|\theta|)|$$

$$zBl(\theta) = \text{int}\left(e^{-|\theta|}\right)$$

$$IO(D_v, offset) = \frac{\text{int}\left(\frac{D_v}{offset}\right)}{\text{int}\left(e^{-\text{int}\left(\left|\frac{D_v}{offset}\right|\right)}\right) + \text{int}\left(\frac{D_v}{offset}\right)}$$

$$skewsine(\theta, ph, V, offset) = \sin\left(\pi - \left[\frac{\sin(ph\times frame(\theta)) + zBl(frame(\theta))}{|\sin(ph\times frame(\theta)) + zBl(frame(\theta))|}\right] + \left[\frac{\pi}{2}\times IO(frame(\theta)) + \frac{D_v - offset\times IO(frame(\theta))}{offset - V\times IO(frame(\theta))}\right]\right)$$

$$skewsine(T, Q) = \sin\left(\pi - \left[\frac{\sin(2\pi\times Q\times frame(T)) + zBl(frame(T))}{|\sin(2\pi\times Q\times frame(T)) + zBl(frame(T))|}\right] + \left[\frac{\pi}{2}\times IO(frame(T)) + \frac{\left|\left[\frac{\pi}{Q}\right] - T\right| - [0.1]\times IO(frame(T))}{[0.1] - \left[\frac{\pi}{Q}\right]\times IO(frame(T))}\right]\right)$$

Equation set 3.15: The custom trigonometric function *skewsine()* was defined in terms of angle (θ), phase (*ph*), "vortex" value (*V*) and offset, as well as the internal function components *frame()*, the distance from the "vortex" value (*Dv*), *zBl()* and *IO()*. The function *zBl(x)* guards against division by zero. *IO()* is a custom function that negates its coefficients by assuming the value of zero depending on input (when the quotient of D_v and *offset* is less than 1). The *skewsine()* function transforms tolerance (*T*) values into the appropriate input for the *sine()* function so as to map the equivalent graph regions as indicated above. The graph shape of the penultimate equation in this equation set is shown in the upper plot of Figure 3.25. In the final equation of this equation set, the *skewsine()* function is expressed in terms of *T* and *Q*.

To test the applicability of the *skewsine()* function detailed in Equation set 3.15, we fitted the *skewsine()* function to the $PerDev_{Da}$ values plotted in blue against tolerance (*T*) for singly charged precursors, in Figure 3.24. Here, the first parameter θ took on the *T* value. By adjusting the phase to 2π, the function passed through zero at intervals of 0.5 Da, in accordance with the data. The *V* value is π and the *offset* values were calibrated to a value of approximately 0.1. Applying these parameter values, *skewsine()* was fitted against the $PerDev_{Da}$ data for singly charged precursors, as depicted in Figure 3.26. Figure 3.26 showed that the custom function *skewsine()* was a moderately good fit and

described the data fairly well ($R^2 \approx 0.8859$; mean of the error squared = 2.45×10^{-5}).

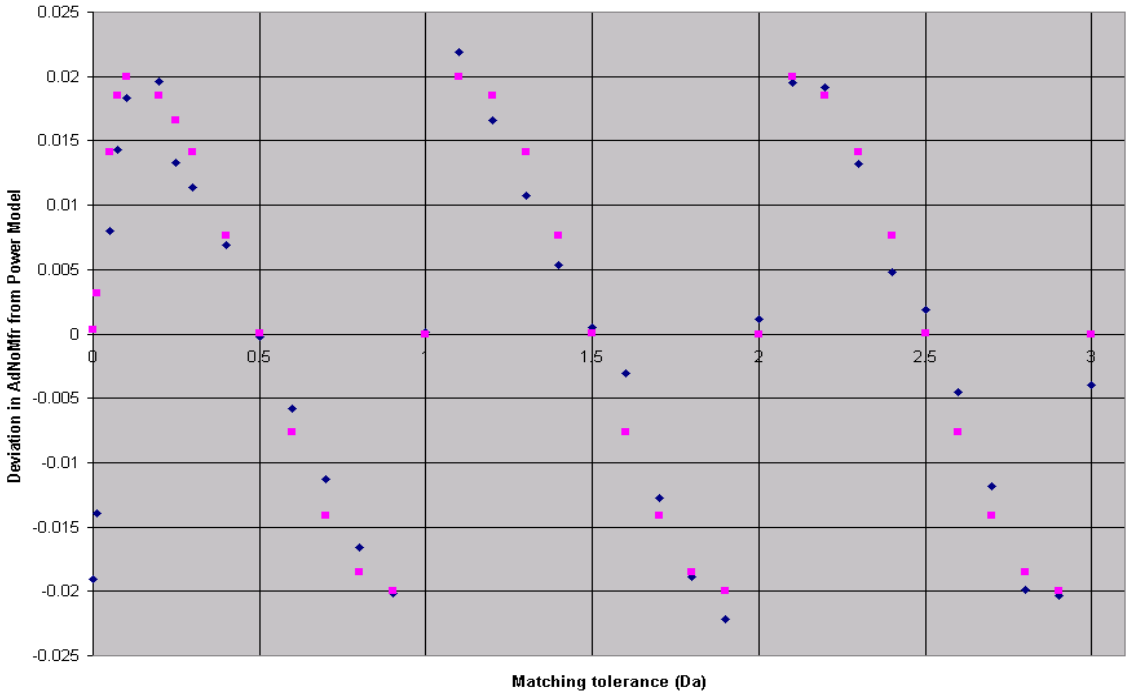
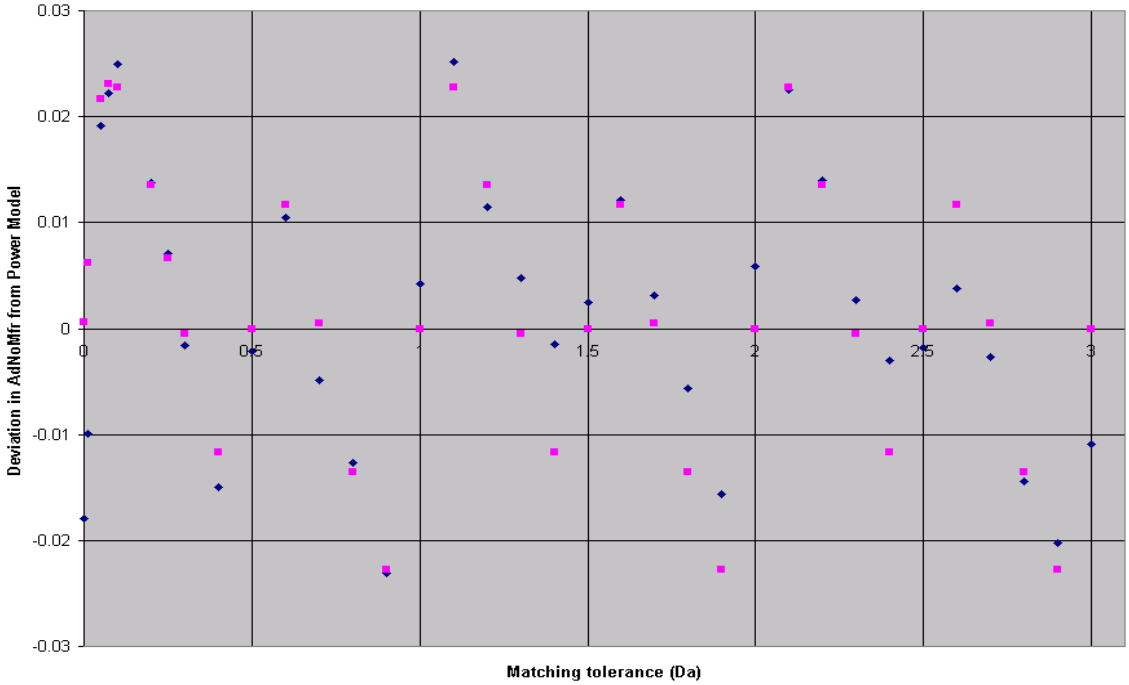


Figure 3.26: Deviations between the derived and calibrated mid-point shape power model and the data for the number of expected peaks tolerance-dependently decoy matched (*P*), are plotted against tolerance (*T*) in Daltons. The *skewsine()* function (see the last equation of Equation set 3.15) was fitted to the data for singly charged precursors.

3.3.2.5.3 Modelling the periodic deviation of P from the mid-point of the data plot for multiply charged precursors

Having modelled $PerDev_{Da}$ for singly charged precursors, we sought to determine the influence of increased precursor charge (*Q*) on $PerDev_{Da}$, and thereby model $PerDev_{Da}$ for multiply charged precursors. To assess the affect of multiple precursor charges, we plotted the $PerDev_{Da}$ values for doubly and triply charged precursors against tolerance (*T*) in Figure 3.27 panels (A) and (B), respectively. Here, in the $PerDev_{Da}$ data for multiply charged precursors, we observed several periodic sets of peaks and troughs, each with a different frequency and amplitude. As previously observed in Figure 3.19 above, an increase in precursor charge increased the number of steps that deviated *P* from the mid-point curve. In Figure 3.26 we could more clearly see that the additional deviations introduced by increasing *Q*, differed in magnitude from those observed for singly charged precursors.

A



B

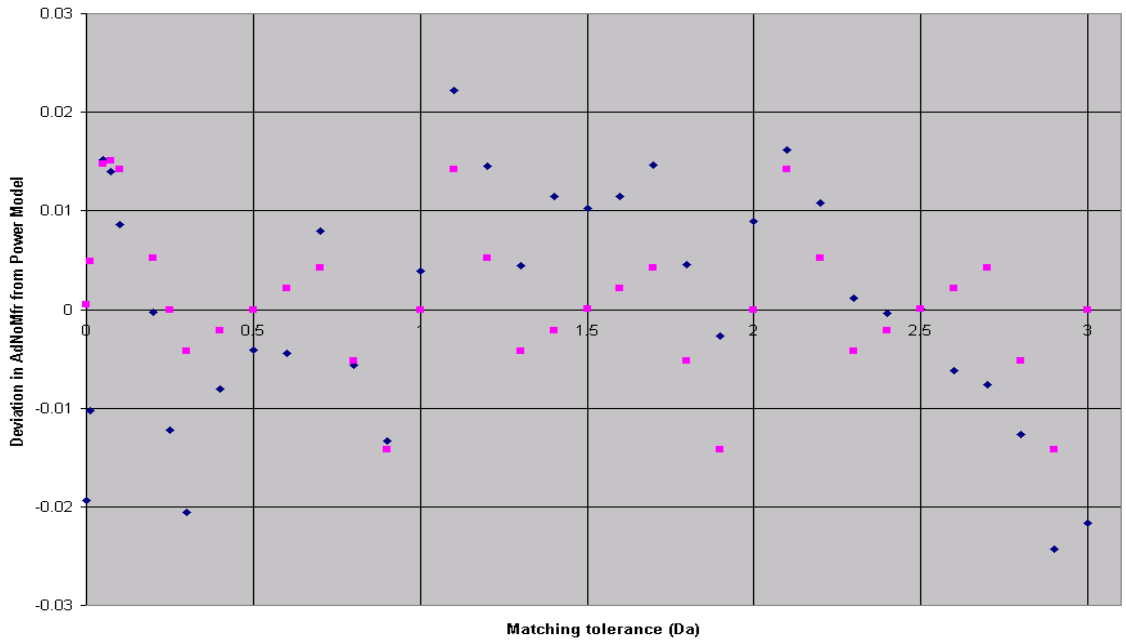


Figure 3.27: Deviation ($PerDev_{Da}$) between the fraction of expected peaks tolerence-dependently decoy matched (P) and the predicted mid-point curve, was plotted in blue against tolerance (T) in Daltons. In panels (A) and (B) the derived sum of *skewsine()* function (see Equation 3.16) was fitted to the $PerDev_{Da}$ data for doubly and triply charged precursors, respectively.

The presence of peaks that assumed different amplitudes with different frequencies was suggestive of the sum of trigonometric functions, each bearing a different coefficient to *sine()* within

skewsine() (see Equation set 3.15). This was consistent with the fact that post-CID fragment ions could assume any charge state up to and including that of the precursor (*Q*). Therefore, where precursors are multiply charged, the fragmentation spectrum would contain fragment ions with a variety of charge states.

Based on the proposed scenario, we constructed Equation 3.16 wherein the $PerDev_{Da}$ value for multiply charged precursors was estimated as the sum of the $PerDev_{Da}$ value estimated by *skewsine()* for each specific fragment charge state (*z*). The first factor in Equation 3.16 estimates what fraction of the total amplitude was attributable to the contribution of a particular charge state (*z*).

$$PerDev_{Da}(Q,T) = \sum_{z=1}^{Q} \frac{z}{\sum_{j=0}^{Q} Q - j} \times MaxAmp(Q) \times skewsine(T,Q)$$

Equation 3.16: The periodic deviation of observed *P* values from the predicted mid-point curve was expressed in terms of precursor charge (*Q*) and tolerance (*T*). Here the function *MaxAmp()* returns the maximum value or amplitude for the data.

We then tested the suitability of Equation 3.16 for the modelling of $PerDev_{Da}$ for multiply charged precursors. Equation 3.16 was fitted to the $PerDev_{Da}$ data for multiply charged precursors in Figure 3.27. This model proved to be a fairly good predictor for singly and doubly charged species (see Figures 3.27 A and 3.27 B). However, the data became increasingly irregular as the charge state increased, and increasingly resisted reliable modelling (refer to Figure 3.27 B) ($R^2 \approx 0.3807$). Equation 3.16 provided a moderate description of the overall shape for triply charged precursors, but failed to accurately model the observed amplitudes.

This does not pose significant problems for the purpose of modelling *P*, as plotted in Figure 3.16, because the periodic deviations became less pronounced with increasing charge state and made a negligible contribution to the value of *P* where *Q* was greater than 2 (see Figure 3.19). Even as the value of *P* increased, the amplitude of periodic deviations remained more or less constant (refer to Figure 3.19 in conjunction with Figure 3.27). In light of this, Equation 3.16 was revised. The resultant Equation 3.17 was used to estimate $PerDev_{Da}$.

$$PerDev_{Da}(Q,T) = \sum_{z=1}^{Q} AmplContrib(Q,z) \times skewsine(T,Q)$$

Equation 3.17: The periodic deviation of observed *P* values from the predicted mid-point curve was

expressed in terms of precursor charge (Q) and tolerance (T). Here the function *AmplContrib()* returns empirically-based amplitude estimates for each fragment charge state (z) where Q is less than 4.

We were unable to extrapolate a generalized model for the trigonometric coefficients across the range of charge states surveyed. However, the periodic deviations only maintained a significant contribution to the overall value for the first three charge states. Since charge state only assumed discrete values, a lookup table for the first three precursor charge states was employed for estimating these trigonometric coefficients in the decoy baseline estimate within the AnchorMS scoring scheme. Here, the periodic deviation term ($PerDev_{Da}$) assumes a zero value for all higher charge states.

3.3.2.6 Composite model for P and D under absolute tolerance

Earlier in this Chapter it was proposed that the value of P, as plotted in Figure 3.19, could be modelled as the sum of the mid-point curve (Equation set 3.13) and the periodic deviation (Equation 3.16) of P from that mid-point curve. Having modelled the two components, we summed the two models to form a composite model for P, described by the formula derived in Equation set 3.17, above.

Next, we sought to test the suitability of their sum for modelling the value of P in terms of Q and T. As shown in Figure 3.28, the composite model for P was fitted to the data for P, plotted against tolerance (T) in Daltons. Here, we can see that the composite model provides a good description of the data for P in terms of T and Q (R^2 = 0.9941, mean of the error squared = 1.64×10^{-4}).

The derived equation of Equation set 3.18, below, summarized the modelling of P and provided an equation for estimating its value, given T and Q. The last line of Equation set 3.10 defines the mean number of decoy matches due to tolerance-dependent decoy matching (D_T) in terms of P, Q and L. By substituting the derived model for P into the derived equation of Equation set 3.10, we obtained an equation that described D_T in terms of T, Q and L. By combining this derived and calibrated model for D_T with the model, *G()*, for the number of exact-mass decoy matches (D_E), we obtained a general equation estimating the total number of decoy matches, based solely on the tolerance (T), precursor charge (Q) and precursor length (L). This process is outlined in Equation set 3.19 below.

The final line of Equation set 3.19 is the final equation in the derivation steps, and estimated the value of D where tolerance was measured as absolute tolerance in Daltons. The final line of Equation set 3.19 was also implemented in AnchorMS, to calculate the

expected number of decoy fragments (given T, Q and L) as a decoy matching baseline within the AnchorMS scoring scheme.

$$P = P(Q,T) = P_{Da,\,midpointshape}(Q,T) + PerDev_{Da}(T,Q)$$
$$P(Q,T) = Equation\ 3.20 + Equation\ 3.24$$
$$P(Q,T) = \left[0.0682 \cdot Q \cdot Tol^{1.1133+0.1346 \cdot Q}\right] + \left[\sum_{z=1}^{Q} AmplContrib(Q,z) \times skewsine(T,Q)\right]$$

Equation set 3.18: A composite equation for estimating the value of P was derived and P was defined in terms of tolerance (T) and precursor charge (Q).

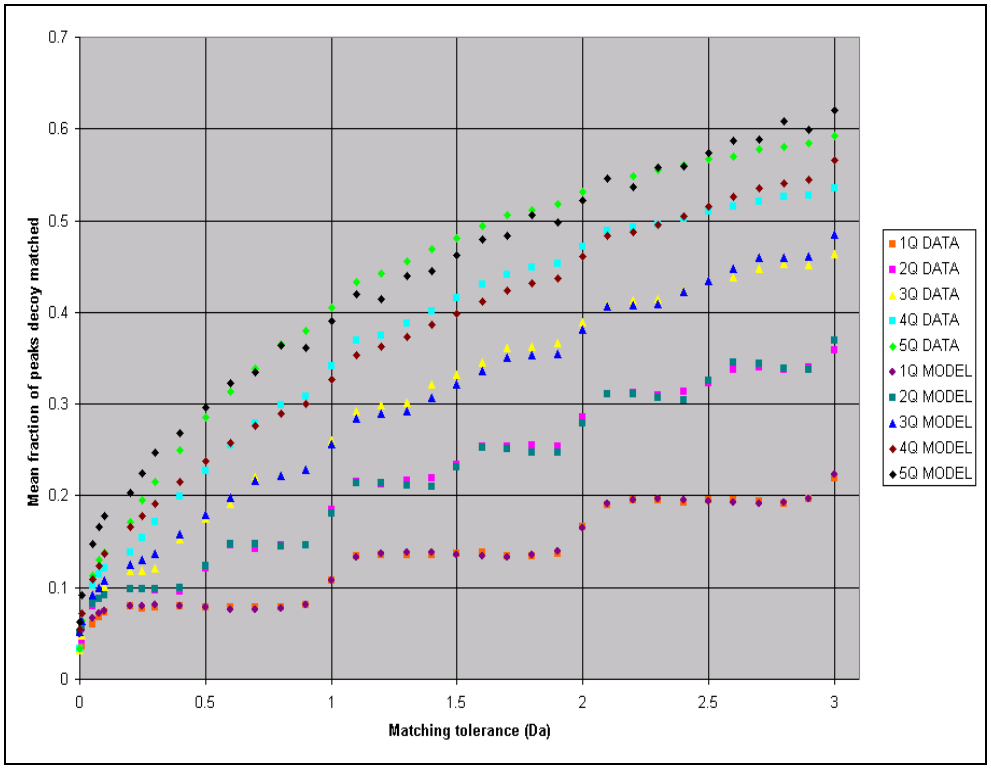


Figure 3.28: The composite model ($P(Q, T)$), calculated as the sum of the mid-point curve and periodic deviation model, was fitted to data for the mean fraction of expected peaks tolerance-dependently decoy matched (P) and plotted against tolerance (T) in Daltons for various precursor charges (Q =(1, 2, 3, 4, 5)).

$$D_{Da} = D_{E,\,Da} + D_{T,\,Da}$$

$$D_{Da} = \text{Equation}\,3.5 + \text{Equation}\,3.10$$

$$D_{Da} = 5.23 \cdot Q \cdot e^{-5.4833 \times e^{-0.4938 \times L}} + \left[\begin{array}{l} Q \cdot \left[0.0281 \cdot L^2 + 5.1457 \cdot L - 15.397 \right] \\ \times P(Q,T) \leftarrow Equation\_3.24 \end{array} \right]$$

$$D_{Da} = Q \cdot \left[\begin{array}{l} 5.23 \cdot e^{-5.4833 \times e^{-0.4938 \times L}} + \\ 0.0281 \cdot L^2 + 5.1457 \cdot L - 15.397 \end{array} \right] \times \left[P_{\text{mid-curve shape, Da}} + PerDev_{Da} \right]$$

$$D_{Da} = Q \cdot \left[\begin{array}{l} 5.23 \cdot e^{-5.4833 \times e^{-0.4938 \times L}} + \\ 0.0281 \cdot L^2 + 5.1457 \cdot L - 15.397 \end{array} \right] \times \left[\begin{array}{l} 0.0682 \cdot Q \cdot T^{1.1133 + 0.1346 Q} + \\ \sum_{z=1}^{Q} AmplContrib(Q,z) \\ \times skewsine(T,Q) \end{array} \right]$$

Equation set 3.19: An equation was derived for the number of decoy matches (D) under absolute tolerance (D_{Da}), which was expressed therein in terms of tolerance (T), precursor charge (Q) and precursor length (L).

3.3.3 Modelling D where relative tolerance (ppm) is applied

In the course of this Chapter we have sought to define a model that was able to predict the number of decoy matches (*D*) and adequately describe the changes in *D* in terms of precursor length (*L*), precursor charge (*Q*) and tolerance (*T*). This was accomplished in the previous section of this Chapter, as demonstrated in Figure 3.28 and summarized in Equation set 3.19. However, the model defined in Equation 3.26 and fitted to the data in Figure 3.28 was only applicable when tolerance (*T*) was measured as an absolute value, in Daltons.

In the steps leading to the derivation of Equation set 3.19, the number of decoy matches (*D*) was described as being equal to the sum ($D = D_T + D_E$) of the number of decoy fragments resulting from exact-mass decoy matching (D_E) and from tolerance-dependent decoy matching (D_T). In equation 3.7, D_E was successfully modelled using the function *G()*, which was equally applicable to both absolute and relative tolerances, since D_E was effectively tolerance independent. For D_T, it could also be shown that there existed a function form, namely a 2nd degree polynomial, that could describe the relationship between D_T and tolerance (*T*) (see Equation set 3.9). Nonetheless, an examination of Figure 3.14 revealed that the specific 2nd degree polynomial functions which successfully fitted the D_T data plotted in Figure 3.14, differed from one another depending on whether the applied *T* was measured as an absolute (in Da) or relative (in ppm or ppb) tolerance.

Based on the observed difference in the D_T data obtained through the application of *T* as an absolute or as a relative tolerance measure, we decided to separately model D_T values obtained through application of absolute ($D_{T,\ Da}$) and of relative ($D_{T,\ ppm}$) tolerances. Following on from the mutually relevant Equation set 3.9, we set out trying to model $D_{T,\ ppm}$ in terms of *L*, *Q* and *T*, measured in ppm, and describe this process in the section below.

In Figure 3.14, the sum of *G()* and a 2nd degree polynomial was fitted to observed *D* values plotted against precursor length (*L*), for various precursor charges (*Q*) and tolerances (*T*). Here, $D_{T,\ ppm}$ values, where relative tolerance was applied, were best fitted by polynomial functions where the coefficient to *L*, represented as *d* in Equation set 3.9, was equal to zero. Therefore, the equation to specifically describe $D_{T,\ ppm}$ could be simplified as shown in Equation 3.20. However, as was seen in Figure 3.14, the rate of change in $D_{T,\ ppm}$ with respect to *L* varied between the data plots and was thus dependent on the values of *T* and *Q*.

$$D_{T,\,pmm} = f.L^2 = f(Q,T)\cdot L^2$$

Equation 3.20: $D_{T,\,ppm}$ was defined in terms of precursor length (L) and the parameter function *f()*.

In order to facilitate further investigation of the factors influencing the value of *f()*, we generated a data set of valid *f* values by fitting Equation 3.20 to a number of data plots where $D_{T,\,ppm}$ was plotted against *L* for a variety of Q and *T* values. In essence, this was an extension of the data fitting previously performed for Figure 3.14.

*3.3.3.1 Modelling the effect of precursor charge (***Q***) on* f()

We next sought to investigate the relationship between *f()* and precursor charge (*Q*) by plotting the fitted *f()* values against *Q* for a variety of tolerances (*T*), as shown in Figure 3.29 below. Figure 3.29 confirms that *f()* indeed depended on both *Q* and *T*. In Figure 3.29, *f()* appeared to increase in an approximately linear fashion with respect to precursor charge (*Q*), where the gradient of the linear increase depended on *T*. As a preliminary test of this linearity, linear equations were fitted to the data plots in Figure 3.29. These fitted line equations all display y-intercepts close to the origin, suggesting that a similar fit could be obtained by considering *f()* to be directly proportional to *Q* and using, instead, line functions that pass through the origin. These linear models provided a moderately good approximation of the data shown in Figure 3.29 (mean R^2 = 0.9909; mean of the error squared = 2.67×10^{-6}). However, the datum point deviations from the linear model were not stochastically distributed across the range of surveyed precursor charges (*Q*). For example, in all the plots shown in Figure 3.29, the fitted linear equation consistently overestimated *f()* where *Q* equalled 2 or 3. Even though the inclusion of only 3 *Q* values in Figure 3.29 might account for this, we felt that this cast some doubt on whether the relationship between *f()* and *Q* was best described using a linear model.

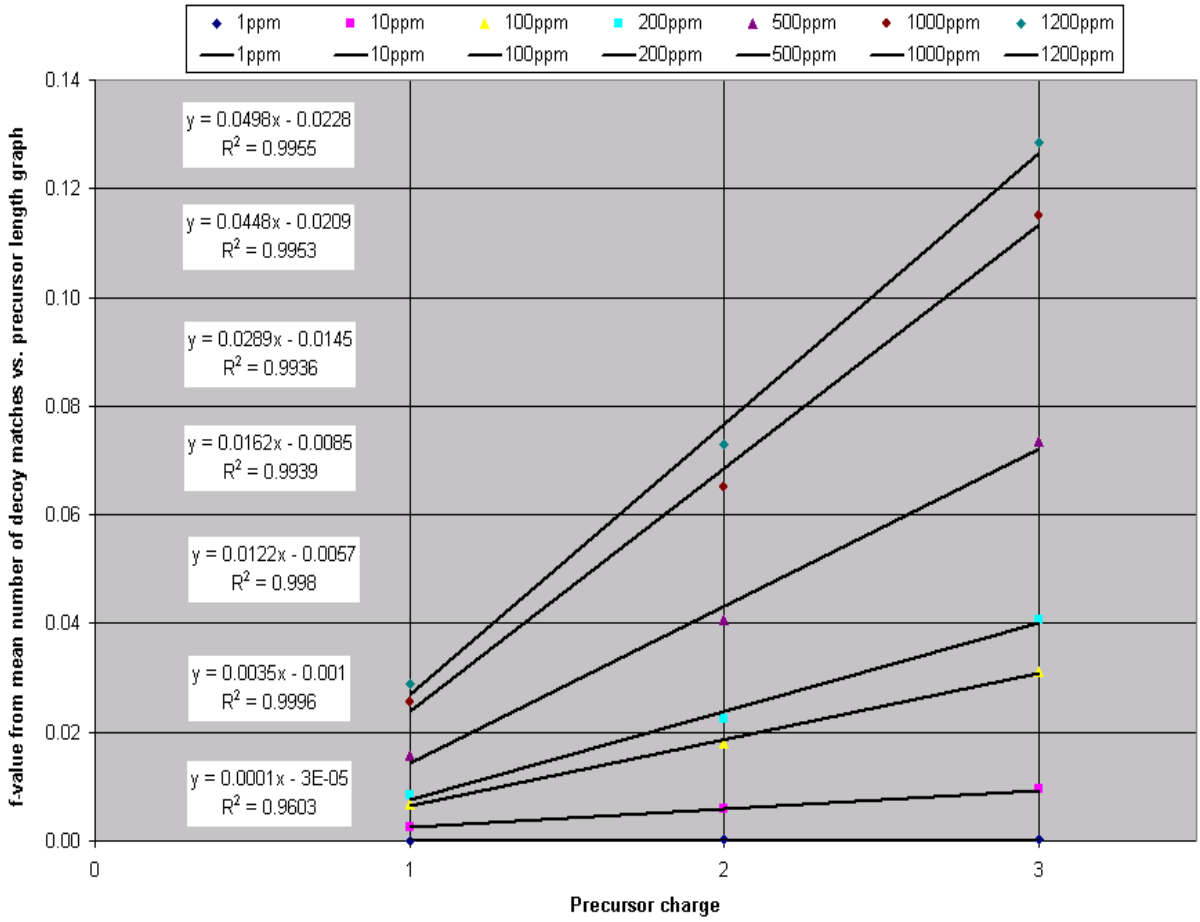


Figure 3.29: Calibrated *f* parameter values were plotted against precursor charge (*Q*) for multiple relative tolerances (*T*). Line equations were also fitted to each data plot.

We decided to test the possibility that *f()* was proportional to an order of *Q* other than 1st order. That is to say, we considered the possibility that *f()* was proportional to *Q* raised to a power other than one. By this notion, the quotient of *f()* and *Q* to the appropriate power (*Qexp*) would be independent of changes in precursor charge (*Q*) (see Equation set 3.21 below).

$$f_{ppm}(Q,T) = f(Q,T,Q\exp,Tterm)$$
$$f_{ppm}(Q,T) = Q^{Qexp(Q,T)} \times Tterm(T)$$

Equation set 3.21: *f()* was expressed in terms of precursor charge (*Q*), as well as the quotient of *f()* and *Q* raised to the power *Qexp* in terms of tolerance (*T*).

In Figure 3.30, *f()* was divided by Q^{Qexp} and plotted against precursor charge (*Q*) for multiple tolerances (*T*). For each plot, *Qexp* was calibrated such that data points in Figure 3.30 remained approximately constant with respect to precursor charge (*Q*). From Figure 3.30, we could see that the calibrated *Qexp* values used for each plot rendered the quotient of *f()* and Q^{Qexp} independent of precursor charge (*Q*) and dependent only on

tolerance (*T*). This result confirmed that *f()* was directly proportional to Q^{Qexp} and that the coefficient of Q^{Qexp} in the second line of Equation set 3.21 was, indeed, dependent on tolerance (*T*). Across the surveyed *Q* values, all *Qexp* values were above 1.2, confirming that the increase in *f()* value with respect to *Q* observed in Figure 3.29, exceeded linearity.

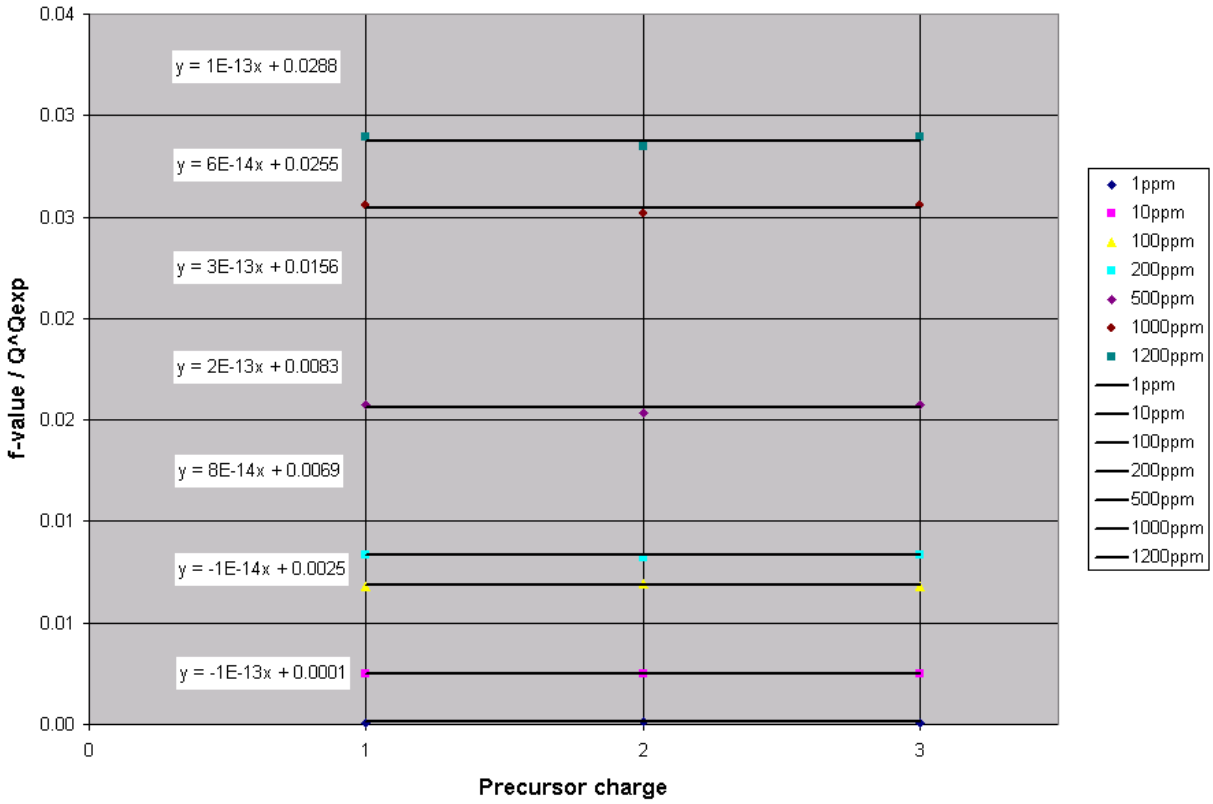
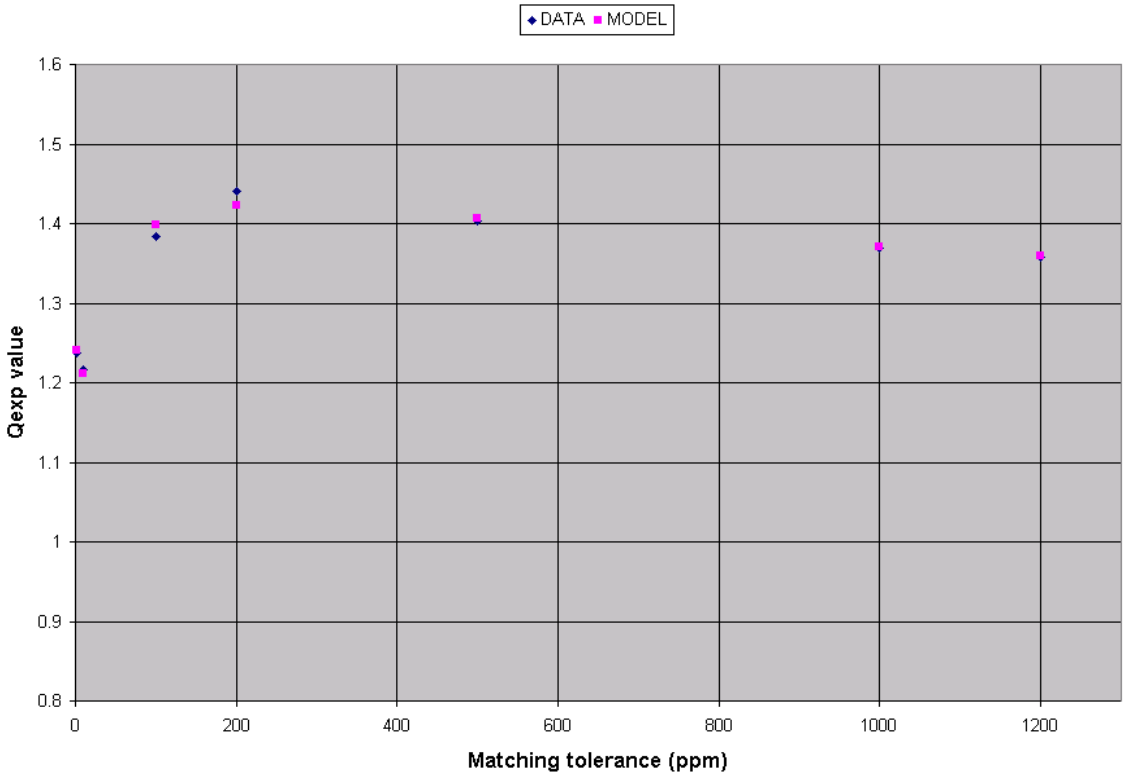


Figure 3.30: The quotient of the calibrated *f()* parameter value and the precursor charge (*Q*) raised to the power of parameter *Qexp,* was plotted against precursor charge (*Q*) for multiple tolerances (*T*). For each tolerance value the *Qexp* parameter value was adjusted to yield an approximately horizontal graph, independent of precursor charge (*Q*).

We then wondered if the calibrated *Qexp* values used for Figure 3.30 were at all dependent on tolerance (*T*). To ascertain this, the calibrated *Qexp* values were plotted against tolerance (*T*), as depicted in Figure 3.31 A, below. Figure 3.31 A showed that Qexp was indeed dependent on tolerance (*T*). *Qexp* was especially responsive to changes in *T* for small *T* values, but became somewhat insensitive to changes in *T* for large values of *T*.

A



B

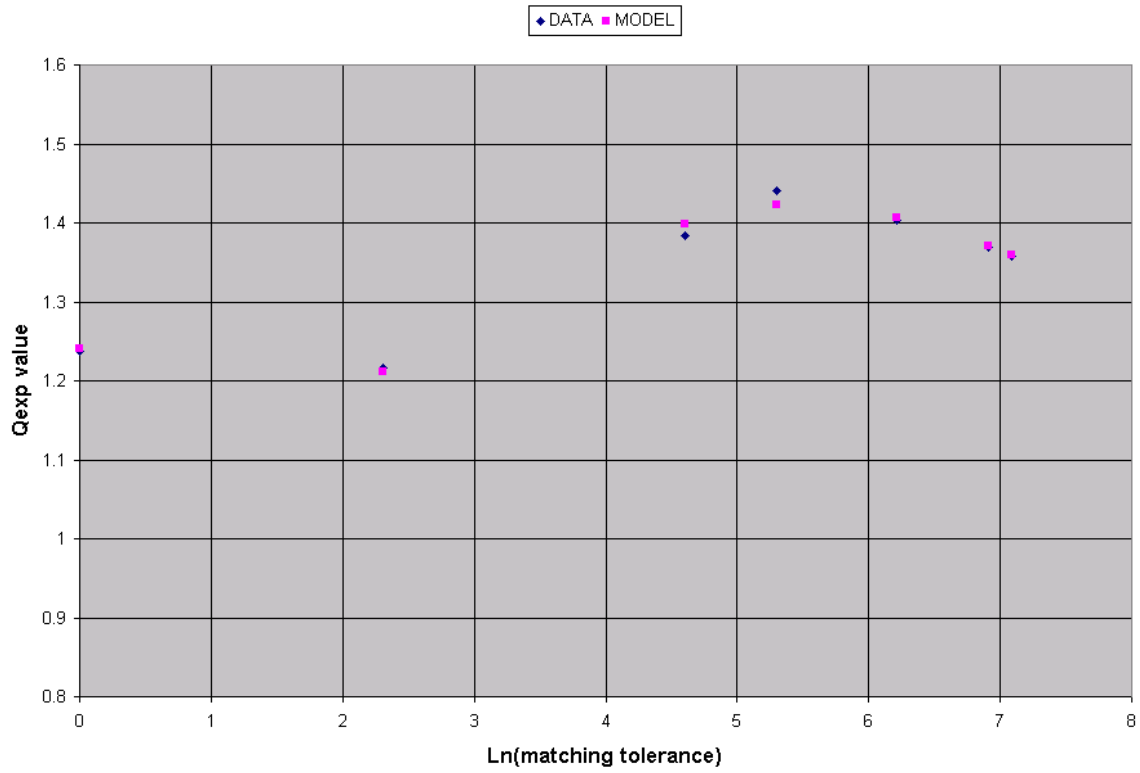


Figure 3.31: The calibrated *Qexp* values were plotted against tolerance (*T*) and the natural logarithm of tolerance (*Ln(T)*) in panels (A) and (B), respectively. The derived model for *Qexp*, from Equation set 3.22, was fitted to the plotted data.

$$Qexp(T) \approx Qexp(T, Qexp0, Qexp1, Qexp2, Qexp3)$$
$$Qexp(T) \approx Qexp0 + Qexp1 \cdot Ln(T) + Qexp2 \cdot \sin(Qexp3 \cdot Ln(T))$$
$$Qexp(T) \approx 1.24070 + 0.018683 \cdot Ln(T) - 0.083184 \cdot \sin(0.90923 \cdot Ln(T))$$

Equation set 3.22: The exponent parameter *Qexp* was defined in terms of precursor charge (*Q*) and tolerance (*T*).

We then attempted to find a model with which *Qexp* could be modelled in terms of tolerance (*T*). The calibrated *Qexp* values plotted in Figure 3.31 A followed an unfamiliar shape with respect to tolerance (*T*). However, when *Qexp* was plotted against the natural logarithm of *T*, as shown in Figure 3.31 B, the data resembles the curve of a sine function that was aligned along a line with a positive gradient. An equation that summed a sine and a linear term, such as the equation on the second line of Equation set 3.22, would follow a similar shape. To test how well this model described the relationship between *Qexp* and *T*, we fitted it against the data plotted in Figure 3.31. For the best fit, both the amplitude and phase of the sine term were adjusted with calibrated constants (*Qexp2* and *Qexp3* respectively). In Figures 3.31 A and 3.31 B, we can see that this model closely followed the data. The last line of Equation set 3.22 summarized the derived equation for estimating the *Qexp*.

3.3.3.2 Modelling f() *in terms of tolerance (*T*)*

The last line of Equation set 3.21 defined *f()* in terms of *Q* as well as the parameters *Qexp* and *Tterm*. Having modelled *Qexp*, we moved on to modelling the parameter function, *Tterm()*. Since *f()* was modelled as the product of Q^{Qexp} and *Tterm()* (see Equation set 3.21), *Tterm()* could be isolated by dividing *f* values by Q^{Qexp}. To investigate the relationship between *Tterm()* and *T*, this data was plotted against tolerance (*T*) in Figure 3.32, below. Here we see that for very low tolerance values, *Tterm* increased rapidly but shifted into an approximately linear form for larger tolerance values. To model this data, we sought a function that followed a similar shape when plotted. Consequently, we constructed an Equation 3.30 which summed tolerance (*T*) and its natural logarithm. This equation was fitted to the data as shown in Figure 3.32, and found to provide a reasonable description of the data (R^2 = 0.9891, mean of the error squared = 1.55×10^{-5}), accurately estimating *Tterm* both at very low and very high tolerance values. The last line of Equation set 3.23 summarizes the results and defined *Tterm* in terms of tolerance (*T*).

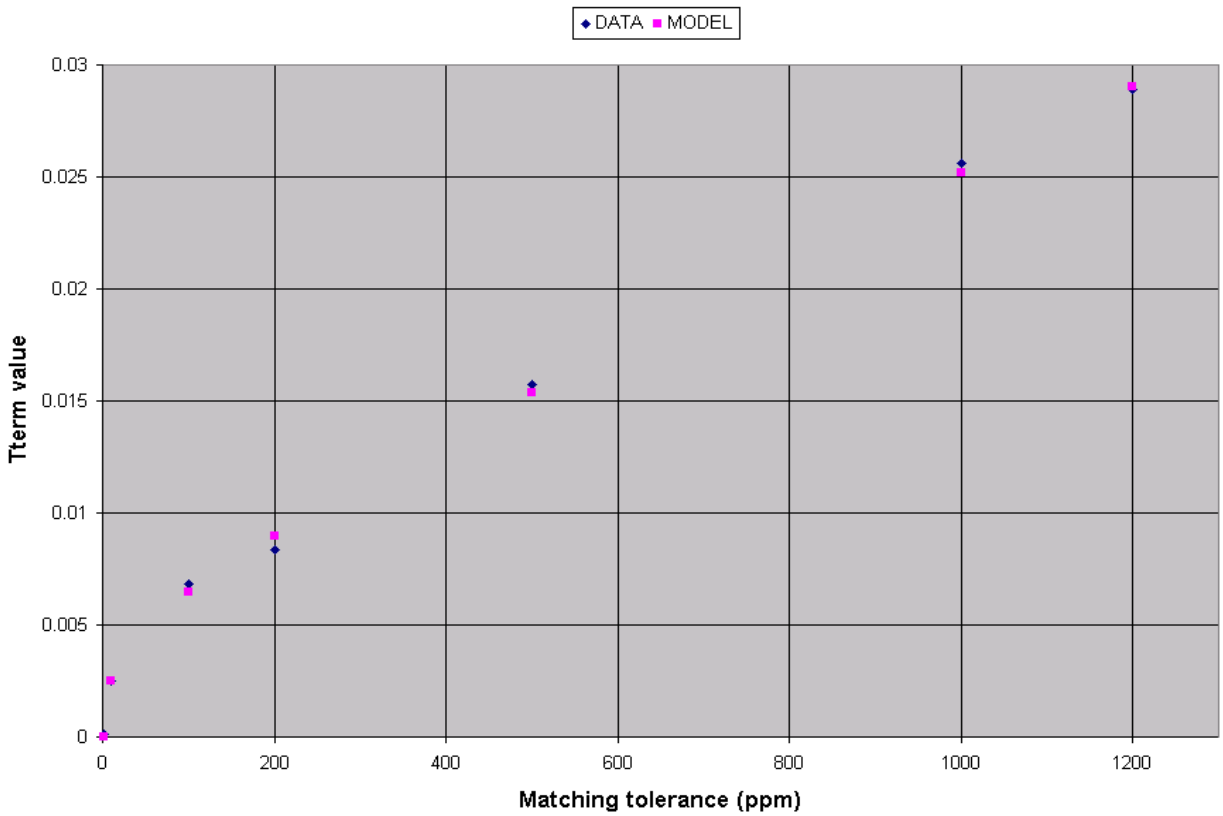


Figure 3.32: The derived model (line 3 of Equation set 3.23) was fitted to the data for *Tterm*, itself derived as the quotient of *f* and precursor charge to the power of calibrated '*Qexp*' (line 1 of Equation set 3.23).

$$1 \quad Tterm(T) \approx \frac{f}{Q\exp}$$

$$2 \quad Tterm(T) \approx Tterm(T, Tt0, Tt1)$$

$$3 \quad Tterm(T) \approx [Tt0 \times Ln(T)] + [Tt1 \times T]$$

$$4 \quad Tterm(T) \approx [0.0010043 \cdot Ln(T)] + [0.000018245 \cdot T]$$

Equation set 3.23: The parameter of *Tterm* was calculated by dividing *f()* by *Qexp* and is defined here in terms of tolerance (*T*) in ppm.

3.3.3.3 Deriving a composite model for f()

Having modelled both *Tterm* and *Qexp*, we sought to incorporate both models into a composite model that could be used to directly model the value of *f()* in terms of precursor charge (*Q*) and tolerance (*T*). In Equation set 3.24, the functions used to model *Qexp* (from Equation set 3.22) and *Tterm* (from Equation set 3.23), respectively, were substituted into the derived equation of Equation set 3.21. The composite model for *f()*, multiplied *Tterm()* with precursor charge (*Q*) raised to the power of *Qexp()* and is shown in Equation set 3.24, below.

In order to determine whether the composite model derived in Equation set 3.24 was a suitable model for *f()*, its estimated values was compared to the data. In Figure 3.33 below, calibrated *f* values were plotted against tolerance (*T*) for multiple precursor charge states (*Q*), and the composite model for *f()*, derived in Equation set 3.24, was fitted to the data plot. Figure 3.33 showed that this composite model supplied a very good approximation (R^2 = 0.9993; mean of the error squared = 8.91×10^{-7}) for the value of the parameter *f*, as defined in Equation 3.20.

$$f_{ppm}(Q,T) = f_{ppm}(Q,T,Q\text{exp}0, Q\text{exp}1, Q\text{exp}2, Q\text{exp}3, Tt0, Tt1)$$
$$f_{ppm}(Q,T) = Q^{Qexp(T,Qexp0,Qexp1,Qexp2,Qexp3)} \times Tterm(T,Tt0,Tt1)$$
$$f_{ppm}(Q,T) = Q^{Equation\ 3.22} \times [Equation\ 3.23]$$
$$f_{ppm}(Q,T) = Q^{Qexp0+Qexp1\cdot Ln(T)+Qexp2\cdot sin(Qexp3\cdot Ln(T))} \times [[Tt0\times Ln(T)]+[Tt1\times T]]$$
$$f_{ppm}(Q,T) = Q^{1.24070+0.018683\cdot Ln(T)-0.083184\cdot sin(0.90923\cdot Ln(T))} \times [[1.0043\times10^{-3}\cdot Ln(T)]+[1.8245\times10^{-5}\cdot T]]$$

Equation set 3.24: A composite model was derived for *f()*, which was defined in terms of precursor charge (*Q*) and tolerance (*T*).

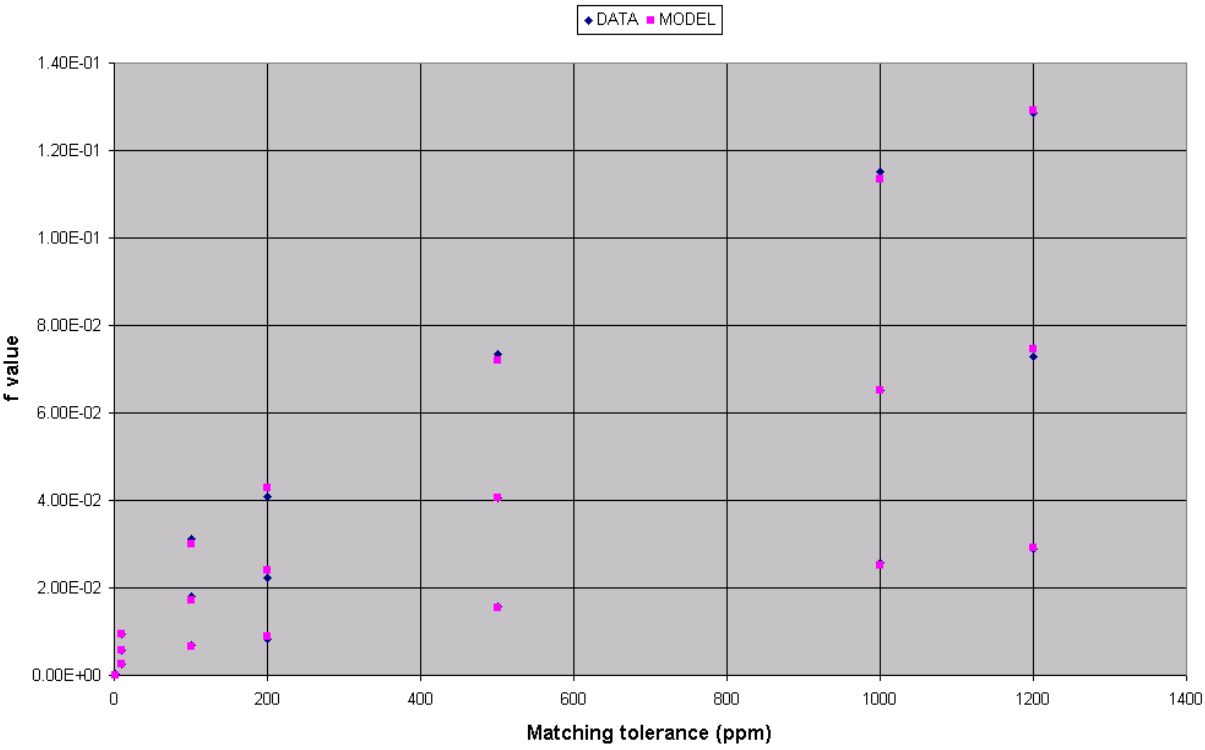


Figure 3.33: Calibrated values for the *f* parameter were plotted against tolerance (T), measured as relative tolerance in ppm, for multiple precursor charge states (*Q*). The composite model for *f* (see the derived equation in Equation set 3.24) was fitted to the plotted data.

3.3.3.4 The complete model for D *where relative tolerances are applied*

With the parameter *f* defined in terms of *T* and *Q*, we returned to Equation 3.20, where *f* was first introduced specifically for relative tolerances. The parameter *f* in Equation 3.20 was substituted with the composite model defined for *f* in Equation set 3.1, providing an equation for $D_{T,\,ppm}$. This was added to the D_E, as defined by Equation 3.9. The result was Equation set 3.25, below, which defines the final equation for the estimation of *D* where relative tolerances (in ppm or ppb) was applied (D_{ppm}).

$$D_{ppm} = D_{T,\,ppm} + D_E$$
$$D_{ppm} = \left[f_{ppm}(Q,T).L^2\right] + \left[G(L,Q)\right]$$
$$D_{ppm} = \left[Last\ line\ of\ Equation\ \ set\ 3.24\right] \times L^2 + \left[Equation\ \ 3.5\right]$$
$$D_{ppm} = Q^{1.2407\ +0.01868\ \cdot L\ -0.083184\ +\sin(0.90923\ +Ln(T))} \cdot \left[1.0043 \times 10^{-3} \cdot Ln(T) + 1.8245 \times 10^{-5} \cdot T\right] \cdot L^2$$
$$+ 5.23 \cdot Q \cdot e^{-5.4833 \cdot e^{-0.4938 \cdot L}}$$

Equation set 3.25: The number of decoy fragment matches due tolerance-dependent decoy matching under relative tolerance (D_{ppm}), was defined in terms of precursor length (L).

Equation 3.32 was also implemented and incorporated into AnchorMS. The implementation of Equation 3.32 estimate the number of expected decoy fragment peaks given a precursor charge, precursor length and relative tolerance measured in ppm. This estimate constitutes the baseline for decoy matching within the AnchorMS scoring scheme against which spectrum matching scores are evaluated.

3.3.4 Sequence-identical precursors differing in cross-link sites

In the simulations described above, threshold or baseline equations for average decoy matching were derived from a large set of decoy fragment spectra. Each decoy fragment spectrum was generated from shuffled sequences. However, in practice, this methodology largely overestimates the degree of false positive matching. Randomly shuffled sequences are still constrained to the same amino acid composition as the original sequence, and may even contain regions identical to the original sequence. As such, the ideal and decoy fragment spectra may share fragments of similar mass due to similar residue composition, even if the fragment sequences differ. Under experimental conditions, alternative experimental spectra matched to a given theoretical spectrum are unlikely to derive from the same sequence.

The notable exception to this is where alternative di-peptide precursor assignments have

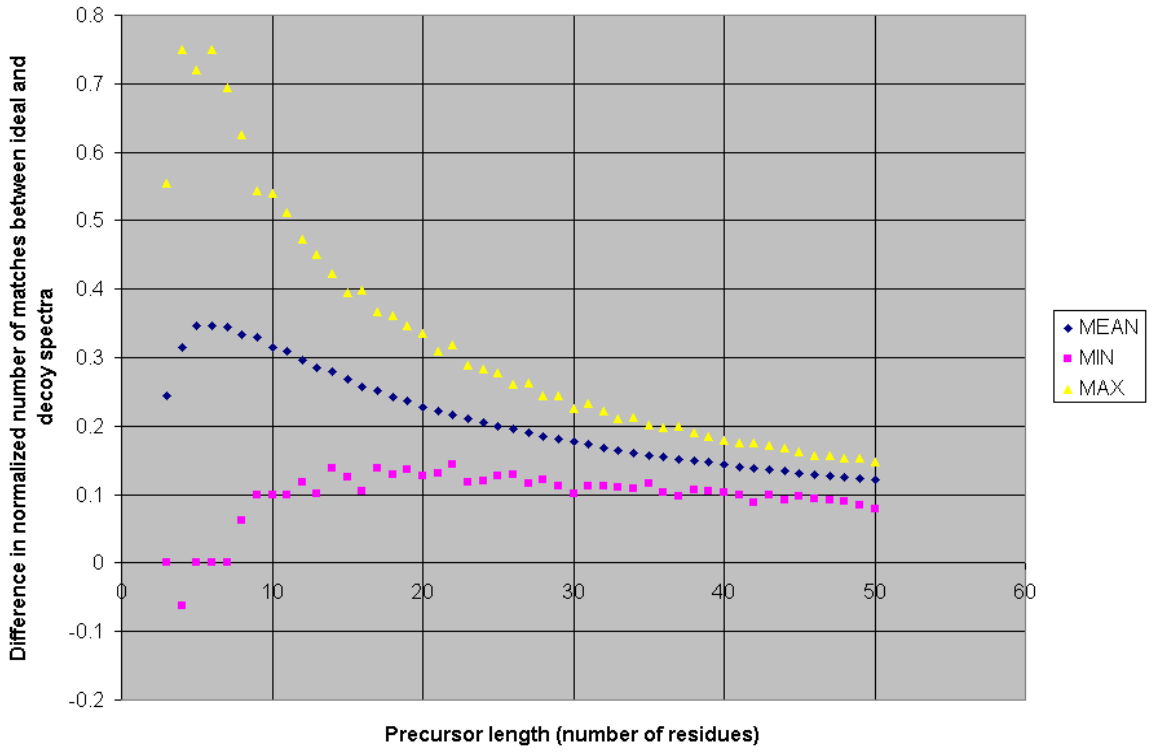
the exact same sequences, and differ only in their cross-link sites. This scenario is highly probable in large precursors containing multiple cross-linkable residues on a peptide or in proteins rich in the amino acids reactive to the cross-linking reagent. For example, this is likely to occur where the commonly used reagent, BS^3, is incubated with lysine-rich histone proteins. Since the sequences are identical in such a case, the resultant fragmentation spectra will share a large number of product ions.

3.3.4.1 The efficacy of the number of fragment peaks matched as a scoring measure

In AnchorMS, the number of fragment peaks matched is effectively used as the primary score for evaluating the closeness-of-fit between observed and predicted MS^2 spectra, and thereby used to determine the most likely assignment for an observed MS^2 spectrum. However, because the MS^2 spectra of sequence-identical di-peptides share a significant portion of their fragment peaks, their evaluation is especially difficult. To test the efficacy of the number of fragment peaks as a scoring measure, we considered its ability to discriminate between an ideal and a decoy spectrum.

Figure 3.34 below compares the number of fragment peaks matched in decoy spectra and ideal spectra, and the difference between the two was plotted against precursor length (L). In Figure 3.34 A, the decoy precursors were sequence shuffled. Here, we can see that, with the exception of very short precursors ($L < 8$ residues), a higher number of fragment peaks was consistently matched in the ideal spectra compared to the decoy spectra. For Figure 3.34 B, decoy precursors were sequence-identical, but the cross-linking sites were randomly repositioned within each strand. Here, the difference in the number of fragment peaks matched in the ideal and decoy spectra approached zero as precursor length increases. In Figure 3.34 B, the minimum difference (plotted in pink) never rose above zero, indicating that there was always the chance that the MS^2 spectrum of a sequence-identical decoy precursor will be indistinguishable from that of an ideal precursor on the basis of the number of fragment peaks matched.

A



B

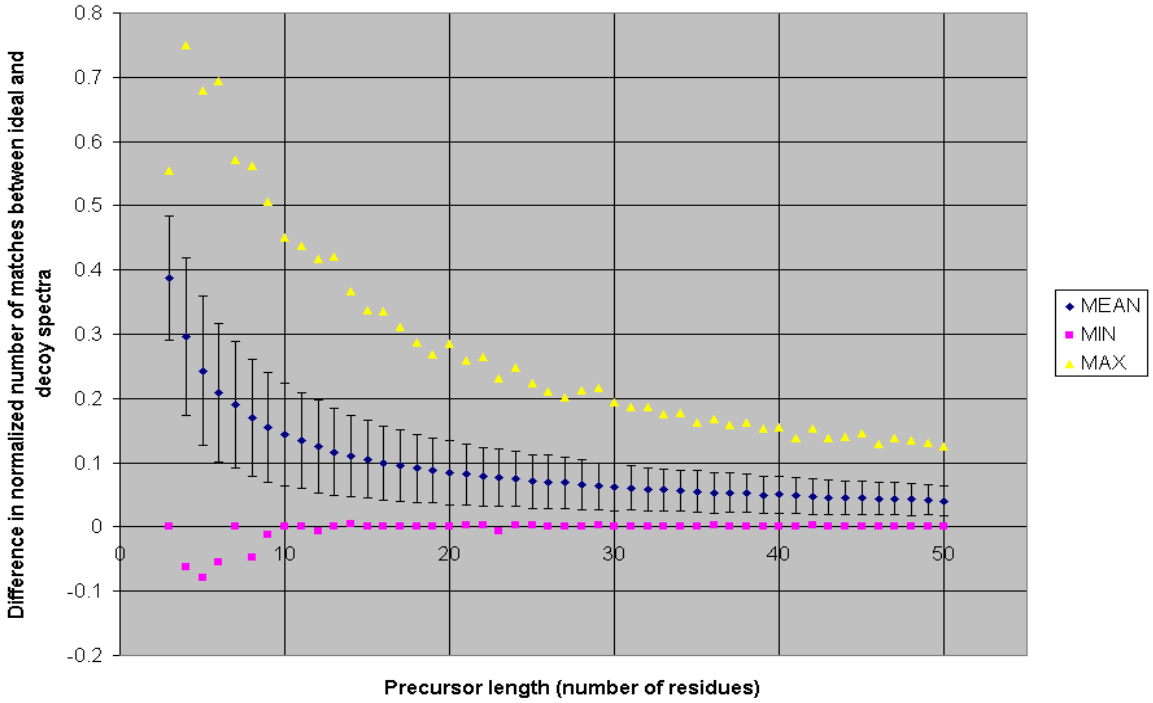


Figure 3.34: Comparison of the separation in ideal and decoy matching levels for sequence-shuffled and sequence-identical precursors, in panels (A) and (B), respectively. The difference between the number of fragment peaks matched in ideal and decoy spectra is plotted against precursor length (*L*) for singly charged precursors at matching tolerances of 5ppm and less.

In short, Figure 3.34 showed that, while the number of fragment peaks matched as scoring measure was suitable for general spectrum matching and assignment, it was insufficient to discriminate between alternative sequence-identical di-peptide assignments.

*3.3.4.2 The efficacy of the Unique Match Count (*UMC*) as a scoring measure for sequence-identical di-peptide precursors*

AnchorMS addresses this problem through the use of a separate scoring measure specifically applied to the comparison of sequence-identical candidate assignments. The Unique Match Count (*UMC*) provides increased discrimination by excluding the shared fragments and focusing only on those theoretical fragment peaks which are unique to that precursor. It is these precursor-unique peaks that distinguish the spectra of each candidate assignment from the alternatives.

Within sequence-identical di-peptides, only the cross-linking site differs. Therefore, any differences that exist between their fragment spectra are due to this difference in cross-linking sites. Figure 3.35 shows why only those fragment ions resulting from fragmentation events between the alternative cross-linking sites will generate fragment peaks that are unique to a particular di-peptide precursor. Therefore, the greater the number of residues that separate alternative cross-linking sites on both peptides, the greater the number of unique fragment peaks that will be observed in fragment spectra of each precursor. The number of residues between alternative cross-linking sites on both strands may be termed the link site distance (*LSD*).

To test the efficacy of *UMC* as a scoring measure, decoy matching for sequence identical decoy precursors was simulated *in silico*. For each of a large number of randomly generated di-peptides, two pairs of cross-linking sites were randomly assigned. The predicted fragment spectrum for each of the two precursors was compared with itself (an ideal match) and with that of the other precursor (a decoy match). In Figure 3.36, the *UMC* values for the ideal and decoy matches are plotted against the link site distance (*LSD*), in yellow and blue respectively.

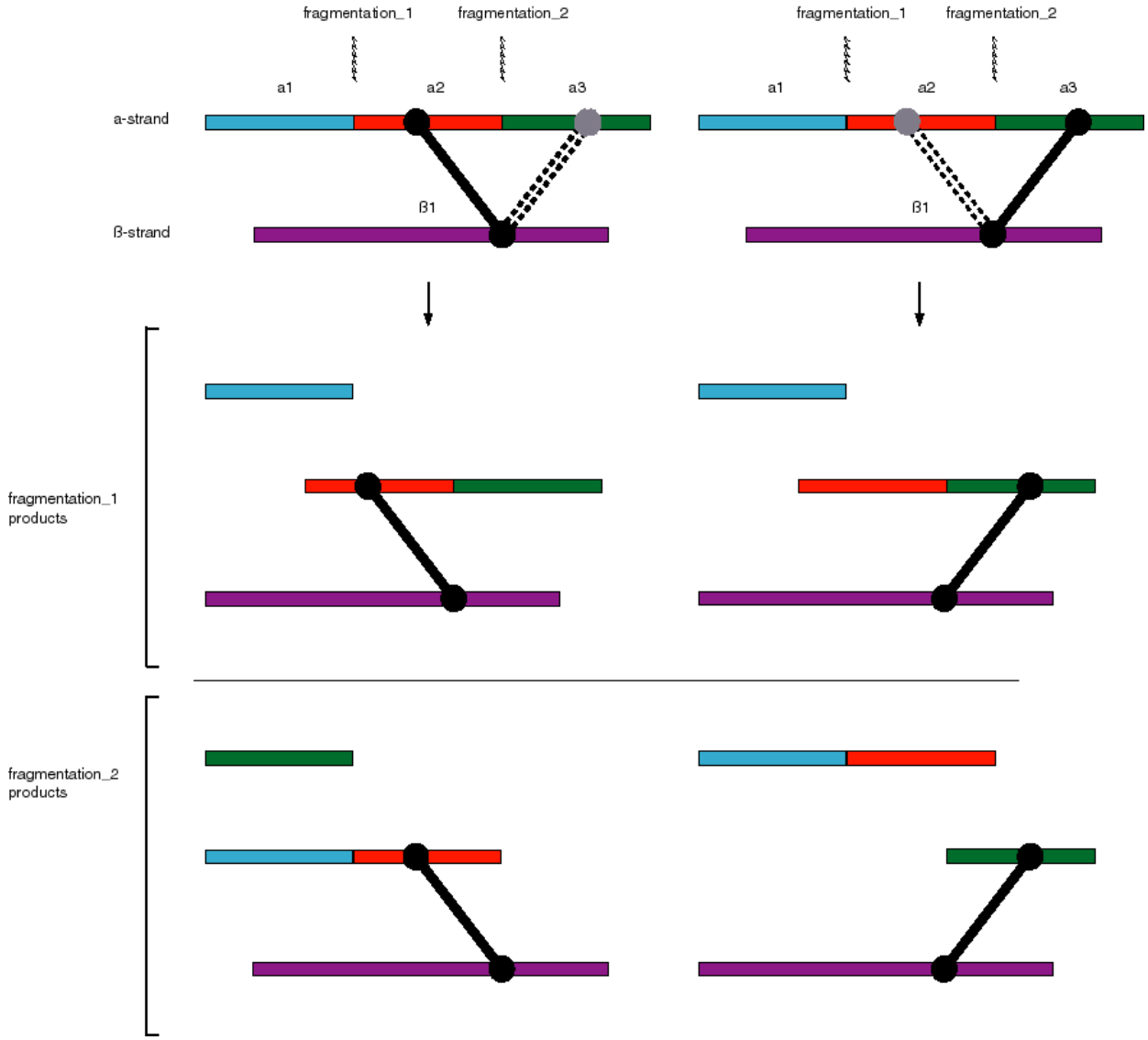


Figure 3.35: In sequence-identical precursors, product ions that are unique in mass and sequence only form when fragmentation occurs between the alternative cross-linking sites (fragmentation_2). Where fragmentation occurs elsewhere in sequence-identical precursors (fragmentation_1), the product ions are identical in sequence and mass.

In contrast to Figure 3.34 B, Figure 3.36 shows a dramatic separation between the plotted score value for ideal and decoy spectra matching. The *UMC* value plotted in Figure 3.36 thus presented an effective discriminator between true and false positives for sequence-identical di-peptide precursors.

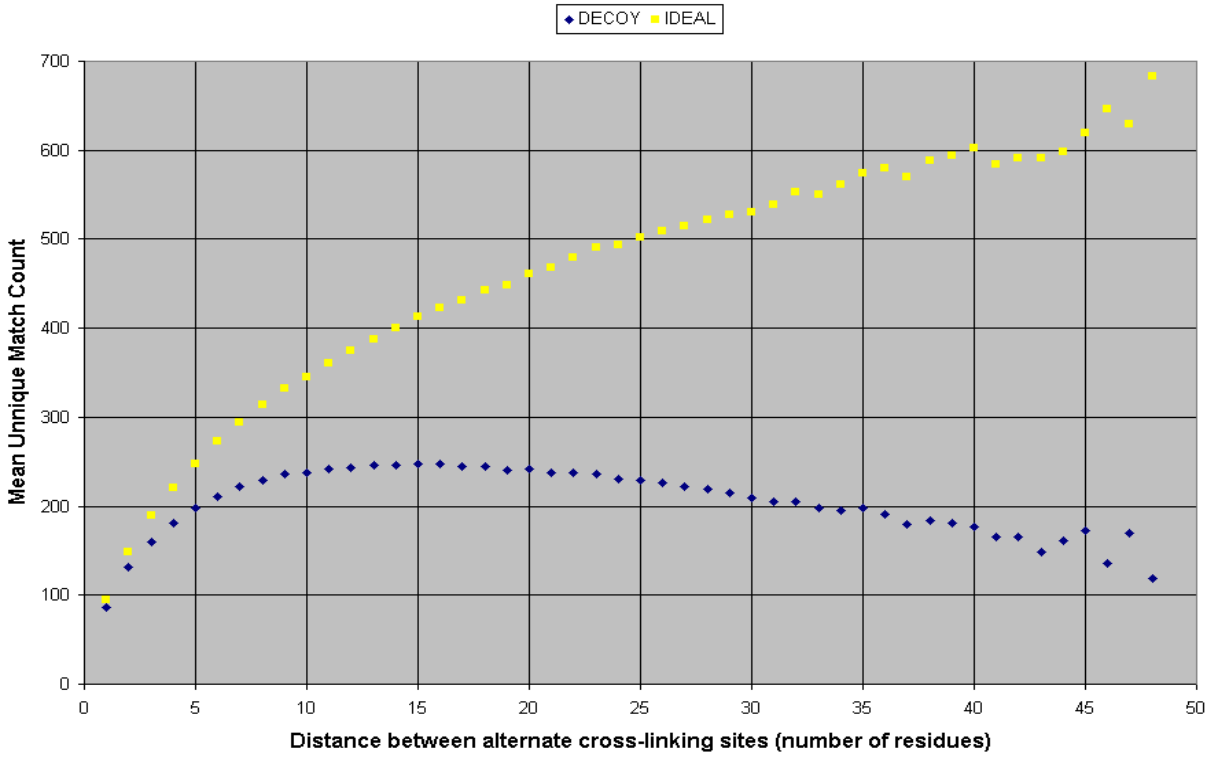


Figure 3.36: Mean Unique Match Count (*UMC*) values from matching of ideal and decoy MS/MS spectra, generated *in silico* over 1,000,000 iterations, were both plotted against link site distance (*LSD*) in yellow and blue, respectively.

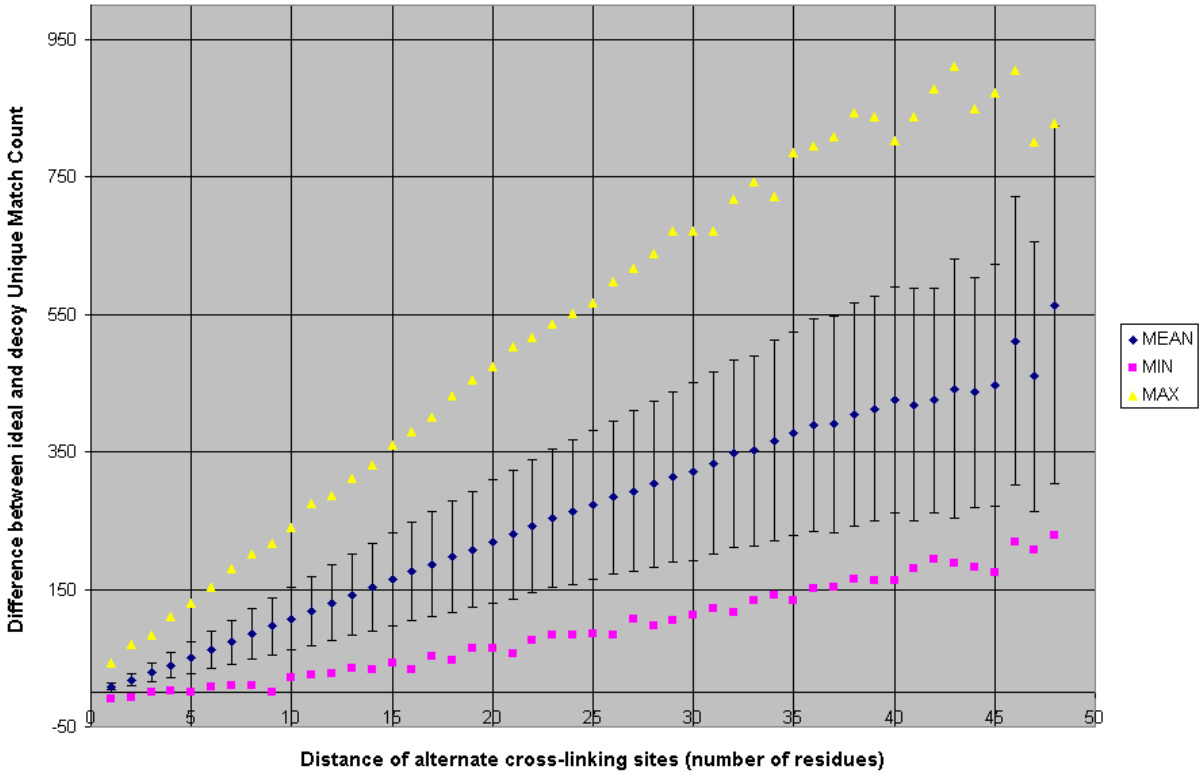
It is worth considering the outliers recorded by the simulation so as to better understand the limits of the *UMC* measure. Figure 3.36 suggested that for very small link site distances, the *UMC* values for ideal and decoy matching might be quite close. To investigate the separation between *UMC* values for ideal and decoy spectrum matches, the minimum and mean difference between the two was plotted against *LSD*, as shown in Figure 3.37. Here we still saw that the minimum difference increases well above zero as *LSD* increased.

Given the large sample size (10^6 iterations), Figure 3.37 confirms that for all inter-site distances of 10 residues or more, true positive assignments were virtually assured for fragments that were unique in both sequence and *m/z* value. For inter-site distances of 1 or 2 residues, Figure 3.37 suggested that it was possible for the fragment spectrum of the decoy assignment to contain more unique matches than that of the true assignment. This may occur at high matching tolerances where unique decoy fragments with *m/z* values only moderately different from that of a theoretical fragment peak, were matched. For example, the masses of several amino acids are within approximately 1 Da of one another,

such as I/L and N, N and D, Q and E. Glutamine and lysine have masses that are even closer (< 0.05 Da). At the largest surveyed matching tolerance (1000 ppm), the absolute matching tolerance is greater than 1Da for precursors larger than 1000 Da. If the fragmentation occurred between two such mass-proximal amino acids, then the resultant fragments may be similar enough in mass to be matched and, because their masses are not identical, they will both be considered as unique fragments.

Even though these outliers are possible, we suspected that they represent rare events. Nevertheless, to roughly gauge the incidence of outlier scenarios, we graphed the density of ideal-decoy *UMC* difference values for various inter-site distances (Figure 3.38). Figure 3.38 A showed a low comparative incidence of negative values for inter-site distance of 1 residue. However, for inter-site distances beyond 1 residue, the incidence of negative values dropped off to a negligible value (Figure 3.38 A – D). We thus conclude that, despite the observed outliers, the *UMC* measure provided a reliable, robust separation between true and false sequence-identical assignments.

A



B

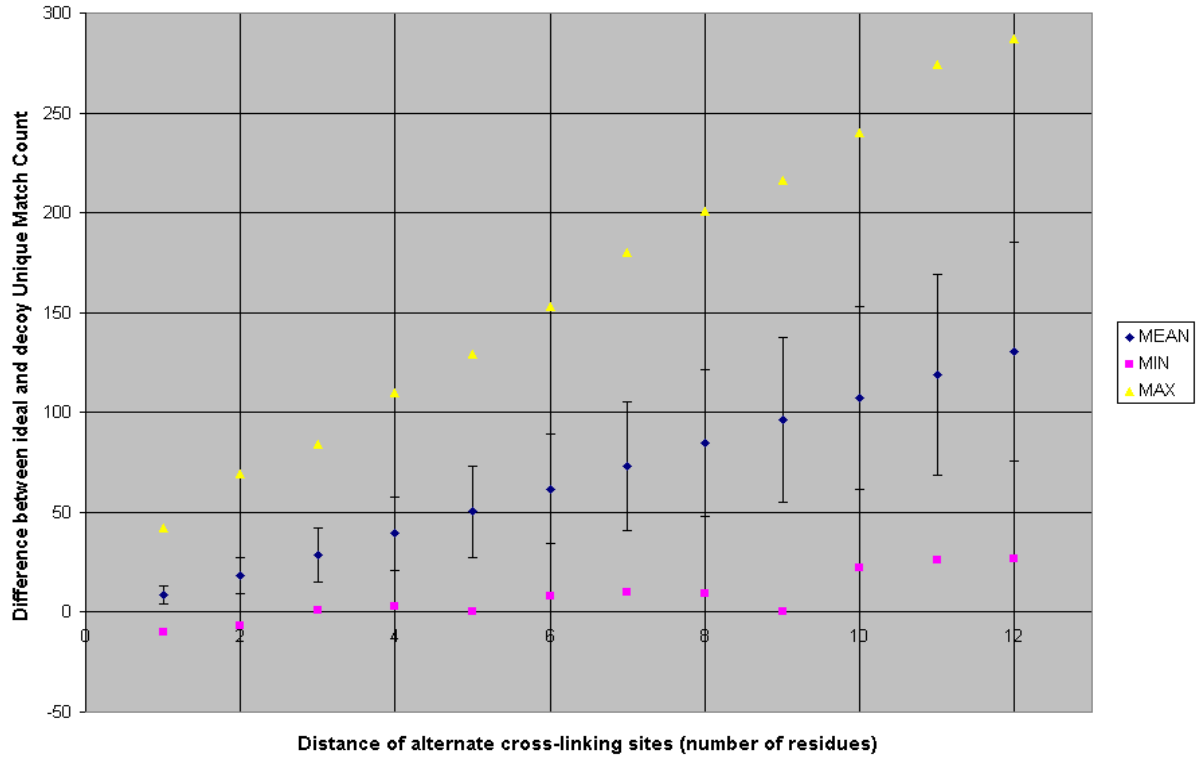


Figure 3.37: The separation between Unique Match Counts from matching ideal and decoy spectra was plotted against the distance between alternative cross-link sites. Panel B "zooms in" on the leftmost portion of Panel A, depicting data up to 12 and 48 residues distance between alternative cross-link sites, respectively.

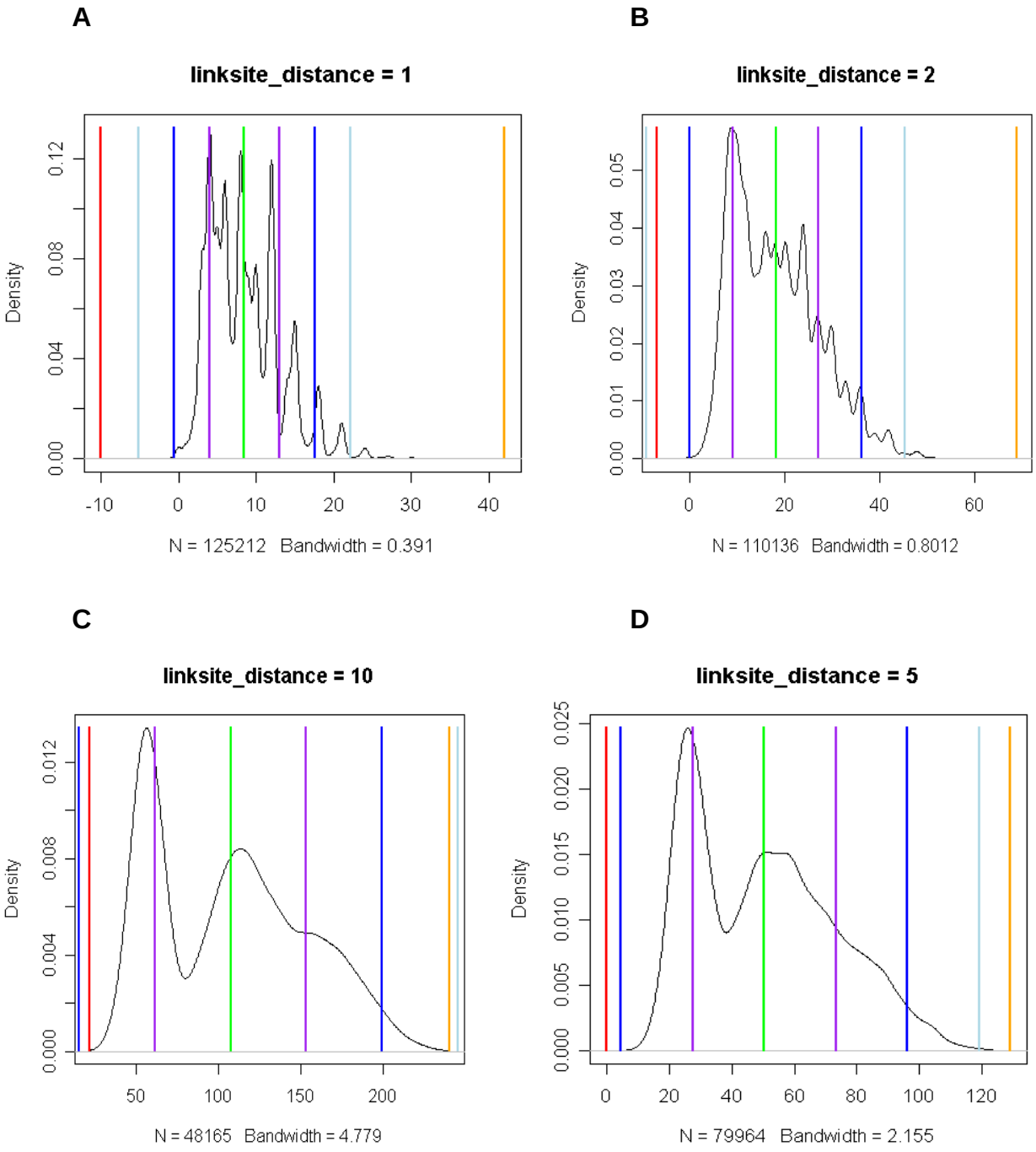


Figure 3.38: Density distributions of 'Unique Match Count' values from matched, *in silico*-generated decoy spectra from sequence-identical di-peptides, showing that the minimum values observed constitute genuine outlier values. Red, green and orange lines indicate the minimum, the mean and the maximum observed values, respectively. Purple, dark blue and light blue indicate 1, 2 and 3 standard deviations from the mean. Panels A-D applies to a link site distance of 1, 2, 5 and 10 residues, respectively.

3.3.5 Implementation of the final false positive matching model in AnchorMS

In the course of this chapter we sought to model the number of decoy matches (*D*) and found it necessary to model *D* differently for absolute (Da) and relative (ppm) matching tolerance respectively. Ultimately, these modelling attempts culminated in the derivation and calibration of two distinct equations for the estimation of *D* in each case (D_{Da} and D_{ppm}, respectively). These two final equations were defined in Equation sets 3.26 and 3.32 above and are revisited in Equation set 3.26 below, by way of summary. The equation for *D* under absolute tolerance (D_{Da}) also included the custom, skewed trigonometric function *skewskine()*, which was represented here in several parts – separating sub-functions *IO()* and *zBl()* – due its complexity. *AmplContrib()* returns hard-coded values stored locally.

$$D_{Da} = Q \cdot \left[\begin{array}{l} 5.23 \cdot e^{-5.4833 \times e^{-0.4938 \times L}} + \\ 0.0281 \cdot L^2 + 5.1457 \cdot L - 15.397 \end{array} \right] \times \left[\begin{array}{l} 0.0682 \cdot Q \cdot T^{1.1133+0.1346 \cdot Q} + \\ \sum_{z=1}^{Q} AmplContrib(Q,z) \\ \times skewsine(T,Q) \end{array} \right]$$

$$skewsine(T,Q) = \sin\left(\pi - \left[\frac{\sin\left(2\pi \times Q \times \mathrm{mod}\left(T, 2\pi Q^{-1}\right)\right) + zBl(T)}{\left|\sin\left(2\pi \times Q \times \mathrm{mod}\left(T, 2\pi Q^{-1}\right)\right) + zBl(T)\right|} \right] + \left[\frac{\pi}{2} \times IO(T) + \frac{|\pi Q^{-1} - \mathrm{mod}\left(T, 2\pi Q^{-1}\right)| - 0.1 \cdot IO(T)}{0.1 - \pi Q^{-1} \cdot IO(T)} \right] \right)$$

$$IO(\theta) = \frac{\mathrm{int}\left(10 \cdot |\pi Q^{-1} - \mathrm{mod}\left(T, 2\pi Q^{-1}\right)|\right)}{\mathrm{int}\left(e^{-\mathrm{int}\left(\mathrm{mod}\left(T, 2\pi Q^{-1}\right)\right)}\right) + \mathrm{int}\left(10 \cdot |\pi Q^{-1} - \mathrm{mod}\left(T, 2\pi Q^{-1}\right)|\right)}$$

$$zBl(T) = \mathrm{int}\left(e^{-\left|\mathrm{mod}\left(T, 2\pi Q^{-1}\right)\right|} \right)$$

$$D_{ppm} = Q^{1.2407 + 0.0187 * L - 0.0832 + \sin(0.9092 + Ln(T))} \cdot \left\lfloor 1.0043 \times 10^{-3} \cdot Ln(T) + 1.8245 \times 10^{-5} \cdot T \right\rfloor \cdot L^2 + 5.23 \cdot Q \cdot e^{-5.4833 \cdot e^{-0.4938 \cdot L}}$$

Equation set 3.26: Two equations derived in Chapter 3 to estimate the number of decoy matches (*D*), where tolerance is absolute (D_{Da}) and where tolerance is relative (D_{ppm}), and each expressed in terms of tolerance (*T*), precursor charge (*Q*) and precursor length (*L*).

These two equations were directly implemented in as Python code in the Scoring.py module of AnchorMS. Where the user supplies a relative tolerance value, the equation for D_{ppm} is used and where the user supplies an absolute tolerance value, the equation for D_{Da} is used. The values of the equation constants are stored in a separate module,

Score_Constants.py, and imported into Scoring.py. Together, these two implemented equations enable AnchorMS to estimate the number of fragment matches likely to be observed if a spectrum match is false and provides a baseline for quantitatively interpreting the number of matches observed. Where the number of matches falls below this false-matching baseline, the match is regarded as false.

3.4 Discussion and Conclusions

In the absence of suitable, available experimental data for calibration, an alternative method was required to derive a decoy-matching baseline. In this chapter we presented the use of *in silico* simulations to describe and model MS^2 matching of decoy peptide precursors. We further described the mathematical and graphical derivation of a set of equations that constituted the decoy-matching baseline implemented in AnchorMS, as a score threshold for confident spectrum assignments.

In practice, the AnchorMS implementation of the peptide fragmentation model upon which the *in silico* matching simulations were based, gave rise to the simulation data used to calibrate the decoy-matching baseline in the AnchorMS scoring scheme. To some extent, this methodology posed a theoretical problem, being circular in its calibration. Indeed, this calibration does rely on the internal consistency of the AnchorMS implementation and the scoring scheme. However, at its heart this calibration serves as an exploration of the inherent mass redundancy and decoy matching potential of peptide precursors based on the created combinatorial space.

Here, sequence and cross-link position define what is meant by the assignable identity. We chose to use sequence-shuffling to generate random matching data. The shuffled decoy sequence differed from the original and generated an overall MS/MS spectrum that also differed from the MS/MS spectrum of the original, even if some fragments were shared. However, the residue composition of the decoy sequence and thus the precursor mass of the decoy sequence, remained the same as the original. We viewed this as a more accurate representation of the experimental scenario where alternative di-peptide assignments must have a similar precursor *m/z* value (within the tolerance) to be matched to some observed MS^1 peak. The alternative form of decoy precursor is one where a sequence is generated by an entirely random assignment of arbitrary residues up to the appropriate precursor length. We considered this method unlikely to generate decoy precursors with mass and *m/z* values similar to the original, true positive precursor. Thus,

we considered experimental scenarios where alternative di-peptide assignments share residue composition (effectively sequence-shuffled) to be the most likely and modelled this form of decoy matching as an approximate substitute for experimental levels of false positive matching.

This does not factor in the likelihood of decoy peptide(s) being at all present in the sample. In rigorously clean samples, false positive matches can only arise from peptides and di-peptides which are cleaved from the analyte protein.

In modelling, we followed a simple process: the available data was organized into appropriate subsets and graphed for manual inspection, followed by the interpolation of the hypothesized model function against the data in order to resolve all parameter values. Ideally a separate and independent, but similar, dataset should be used to test the fit of the model calibrated from the original dataset. In modelling, separate calibration and testing datasets are often created by dividing the available data into subsets. The calibrating dataset is typically the larger since larger calibrating datasets improve the accuracy of the parameter values.

We have shown that the number of matches increased with precursor length, introducing a precursor length-dependant bias favouring large precursors. Several software packages, including MS2Assign, SearchXLinks and X!Link (Lee *et al.*, 2007; Schilling *et al.*, 2003; Wefing *et al.*, 2006) use the number of matches without adjusting the primary scoring component for alternative assignments and, in so doing, introduce a bias. The AnchorMS scoring scheme normalizes against length. However, this normalization is only effective for larger precursors. In fact, the normalization process inverts the trend in length-dependent short precursors. Here, the normalized number of matches decreases with increasing precursor size. Beyond this, short peptide precursors have few purely combinatorial sequence alternatives, making them inherently less unique. In all, this leaves the scoring of short precursor assignments somewhat uncertain. For optimum confidence during an AnchorMS analysis, only precursors longer than 10 residues should be considered.

3.5 References

1. Bakhtiar, R. and Nelson, R. W. (2000) Electrospray ionization and matrix-assisted laser desorption ionization mass spectrometry. Emerging technologies in biomedical sciences. *Biochem.Pharmacol.* **59,** 891-905.

2. Boersema, P. J., Mohammed, S., and Heck, A. J. (2009) Phosphopeptide fragmentation and analysis by mass spectrometry. *J.Mass Spectrom.* **44,** 861-878.

3. Gompertz, B. (1825) On the nature of the function expressive of human mortality, and on a new mode of determining the value of life contingencies. *Philosophical Transactions of the Royal Society of London* **115,** 513-583.

4. Hung, C. W., Schlosser, A., Wei, J., and Lehmann, W. D. (2007) Collision-induced reporter fragmentations for identification of covalently modified peptides. *Anal.Bioanal.Chem.* **389,** 1003-1016.

5. Laird, A. K. (1964) Dynamics of tumour growth. *Br.J.Cancer* **13,** 490-502.

6. Lee, Y. J., Lackner, L. L., Nunnari, J. M., and Phinney, B. S. (2007) Shotgun cross-linking analysis for studying quaternary and tertiary protein structures. *J.Proteome.Res.* **6,** 3908-3917.

7. Paizs, B. and Suhai, S. (2005) Fragmentation pathways of protonated peptides. *Mass Spectrom.Rev.* **24,** 508-548.

8. Schilling, B., Row, R. H., Gibson, B. W., Guo, X., and Young, M. M. (2003) MS2Assign, automated assignment and nomenclature of tandem mass spectra of chemically crosslinked peptides. *J.Am.Soc.Mass Spectrom.* **14,** 834-850.

9. Wefing, S., Schnaible, V., and Hoffmann, D. (2006) SearchXLinks. A program for the identification of disulfide bonds in proteins from mass spectra. *Anal.Chem.* **78,** 1235-1241.

Chapter 4

A demonstration of AnchorMS functionality: the analysis of MS^1 and MS^2 datasets

4.1 Introduction

Multi-subunit protein complexes play an important role in many biological processes. The study and determination of the three-dimensional structure of protein complexes and their constituent protein molecules contributes to an understanding of the mechanism of their biological function. Nuclear magnetic resonance (NMR) or X-ray crystallography (XRC) are traditionally used for the determination of protein structure, but are not suitable for very large protein complexes. With increasing frequency, the mass spectrometric (MS) analysis of post-digestion peptides from cross-linked protein complexes have been used to study the structural features of very large protein complexes (Serpa *et al.*, 2012). This approach is termed MS3D. Cross-linked di-peptides are identified in the post-digest mixture. Each cross-link detected indicates that the cross-linked residues are sufficiently proximal in the native structure to allow the molecular arm of the cross-linking reagent to span the distance between them.

The detection of di-peptides requires suitable software for the analysis of the MS^1 and MS^2 spectra. AnchorMS was developed to fulfil this need and identifies di-peptides in a post-digest mixture according to MS^1 and MS^2 data.

In Chapter 2 the features and code of AnchorMS were discussed in detail. However, software must be tested in order to confirm its functionality. Such testing is ideally performed on experimental data. Despite the wealth of bioinformatic databases, a database of mass spectrometric data for cross-linked di-peptides has yet to be made publicly available. Therefore, to test the basic functionality of AnchorMS, a simple theoretical MS dataset was constructed and analysed. We describe each step of the construction of this dataset. We consider the cross-linking of a real-world protein and its subsequent digestion with trypsin. One of the precursors in the constructed MS^1 spectrum was selected and used for the construction of an MS^2 spectrum. Thus, both the MS^1 and MS^2 analysis of AnchorMS was tested to demonstrate proper AnchorMS functionality.

4.2 Materials and methods

4.2.1 Protein sequence and structure

A small protein was required for the construction of the MS dataset. A pleckstrin homology domain from an α-PIX protein in mice was selected for this purpose. The pleckstrin homology (PH) domain consists of approximately 120 residues and is found in a variety of proteins, especially those involved in intracellular signalling or those serving as constituents within the cytoskeleton (Manser *et al.*, 1998; Mayer *et al.*, 1993; Musacchio *et al.*, 1993; Yu *et al.*, 2004). The PH domain described by this structure is taken from an α-PIX protein in mice, specifically Rac/Cdc42, a guanine nucleotide exchange factor (GEF) 6 (UniProtKB ID: Q8K4I3). Here PIX denotes PAK-interacting exchange factor and PAK denotes a p21 activated kinase. This protein plays a role in intracellular signalling. For the purpose of this study, however, the function of the protein is immaterial.

The protein sequence in fastA format (Figure 4.1) and the three-dimensional NMR solution structure (PDB ID: 1V61) were downloaded from the Protein Data Bank (Berman *et al.*, 2003; Rose *et al.*, 2013).

```
>1V61:A|Pleckstrin Homology domain of mouse alph-PIX
GSSGSSGQIL SEPIQAWEGD DIKTLGNVIF MSQVVMQHGA CEEKEERYFL LFSSVLIMLS
ASPRMSGFMY QGKIPIAGMV VNRLDEIEGS DCMFEITGST VERIVVHCNN NQDFQEWMEQ
LNRLTKSGPS SG
```

Figure 4.1: The sequence of the Pleckstrin Homology (PH) domain of mouse α-PIX protein, downloaded from the protein data bank (ID: 1V61).

4.2.2 Cross-linkable residues

The cross-linking reagent bis(sulfosuccinimidyl) suberate (BS^3) is widely used to cross-link lysine residues within a folded protein subunit or protein complex. In Figure 4.2 below the four lysine residues are highlighted.

```
GSSGSSGQIL SEPIQAWEGD DIKTLGNVIF MSQVVMQHGA CEEKEERYFL LFSSVLIMLS
ASPRMSGFMY QGKIPIAGMV VNRLDEIEGS DCMFEITGST VERIVVHCNN NQDFQEWMEQ
LNRLTKSGPS SG
```

Figure 4.2: The sequence of the PH domain of mouse α-PIX protein with the lysine residues highlighted.

The PDB structure file contains the three-dimensional co-ordinates of all atoms in the

protein structure. The spatial co-ordinates of the α-carbon of each lysine in the protein sequence were extracted from the PDB structure file, and is shown in Table 4.1.

	X (Å)	**Y (Å)**	**Z (Å)**
K23 α-carbon	0.459	-11.317	10.065
K44 α-carbon	10.642	13.771	-3.121
K73 α-carbon	0.301	2.957	9.996
K126 α-carbon	-15.246	2.858	1.465

Table 4.1: Three-dimensional co-ordinates of the α-carbon of the lysine residues within the PH domain of the mouse α-PIX protein (PDB ID: 1V61).

4.2.3 Distance constraints for cross-linkable residues

Although all unmodified lysine side chains are chemically reactive to the cross-linking reagent BS^3, two lysines must also be sufficiently close within the protein structure to be cross-linked. To determine which lysine residues in the PH domain were close enough to be cross-linked, the inter-lysine distances were calculated for each. The inter-lysine distance was calculated as the distance between the α-carbons of each residue. The molecular spacer arm of BS^3 has a reach of approximately 11.34 Å, and the length of the lysine side-chain is 6-6.5 Å. Therefore, inter-lysine distances must be within approximately 24 Å to allow cross-linking. The results are summarized in table 4.2 below and the calculation of distances shown in equation 4.1.

Equation 4.1: $Distance = \sqrt{[X_1 - X_2]^2 + [Y_1 - Y_2]^2 + [Z_1 - Z_2]^2}$

	K23	**K44**	**K73**	**K126**
K23	-	30 Å	14 Å	23 Å
K44		-	20 Å	28 Å
K73			-	18 Å
K126				-

Table 4.2: The inter-residue distances (Å) from α-carbon to α-carbon within the NMR solution structure (PDB 1V61) are shown. Green blocks denote possible cross-links and magenta blocks denote impossible cross-links, when using the BS^3 cross-linking reagent.

Table 4.2 shows that only 4 of the possible inter-lysine cross-links are spatially feasible. The inter-residue distances are too far for the BS^3 linker arm to span between lysine 23 and lysine 44, or between lysine 44 and lysine 126. We show the

positions of these lysine residues within the folded protein structure in Figure 4.3.

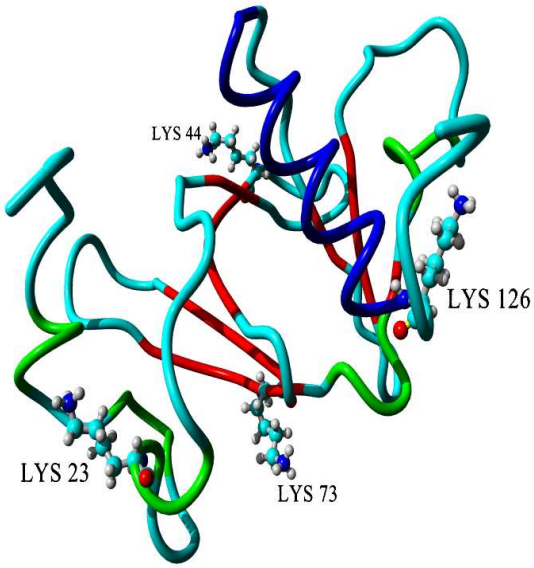


Figure 4.3: A three-dimensional representation of the NMR-derived solution structure for mouse nucleotide exchange factor (PDB ID: 1W61). The positions of the lysine residues are indicated.

4.2.4 Trypsin digestion

After treatment with a cross-linking reagent, the cross-linked protein is typically digested with a protease which cuts the protein strand at specific residues or at specific sequence features. Here we consider the application of the widely used protease, trypsin. Trypsin cleaves the peptide bond immediately on the C-terminal side of lysine (K) or arginine (R) residues (Siepen *et al.*, 2007). We searched through the protein sequence and identified all sites in the protein sequence susceptible to trypsin cleavage, as shown in the sequence in Figure 4.4 below. Trypsin cleavage sites are denoted by underlined residue symbols.

GSSGSSGQIL SEPIQAWEGD DI**K**TLGNVIF MSQVVMQHGA CEE**K**EE**R**YFL LFSSVLIMLS
ASPRMSGFMY QG**K**IPIAGMV VN**R**LDEIEGS DCMFEITGST VE**R**IVVHCNN NQDFQEWMEQ
LN**R**LT**K**SGPS SG

Figure 4.4: The sequence of the PH domain of mouse α-PIX protein. The lysine and arginine residues are underlined and in bold to indicate trypsin recognition sites.

In Figure 4.5, below, we compiled a list of peptides that would result from cleaving the protein at the identified cleavages sites. However, proteases may not always digest the protein with 100% efficiency. Therefore, we considered the possibility of two missed cleavage site per peptide molecule.

0 Missed cleaves (10 peptides):

GSSGSSGQILSEPIQAWEGDDIK

TLGNVIFMSQVVMQHGACEEK

EER

YFLLFSSVLIMLSASPR

MSGFMYQGK

IPIAGMVVNR

LDEIEGSDCMFEITGSTVER

IVVHCNNNQDFQEWMEQLNR

LTK

SGPSSG

1 missed cleave (9 peptides):

GSSGSSGQILSEPIQAWEGDDIKˇTLGNVIFMSQVVMQHGACEEK

TLGNVIFMSQVVMQHGACEEKˇEER

EERˇYFLLFSSVLIMLSASPR

YFLLFSSVLIMLSASPRˇMSGFMYQGK

MSGFMYQGKˇIPIAGMVVNR

IPIAGMVVNRˇLDEIEGSDCMFEITGSTVER

LDEIEGSDCMFEITGSTVERˇIVVHCNNNQDFQEWMEQLNR

IVVHCNNNQDFQEWMEQLNRˇLTK

LTKˇSGPSSG

2 missed cleaves (8 peptides):

GSSGSSGQILSEPIQAWEGDDIKˇTLGNVIFMSQVVMQHGACEEKˇEER

TLGNVIFMSQVVMQHGACEEKˇEERˇYFLLFSSVLIMLSASPR

EERˇYFLLFSSVLIMLSASPRˇMSGFMYQGK

YFLLFSSVLIMLSASPRˇMSGFMYQGKˇIPIAGMVVNR

MSGFMYQGKˇIPIAGMVVNRˇLDEIEGSDCMFEITGSTVER

IPIAGMVVNRˇLDEIEGSDCMFEITGSTVERˇIVVHCNNNQDFQEWMEQLNR

LDEIEGSDCMFEITGSTVERˇIVVHCNNNQDFQEWMEQLNRˇLTK

IVVHCNNNQDFQEWMEQLNRˇLTKˇSGPSSG

Figure 4.5: Peptides produced by digestion of the PH domain of mouse alpha-PIX with trypsin, allowing for up to 2 missed cleave per protein (27 peptides).

1. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEK

2. TLGNVIFMSQVVMQHGACEEKEER

3. MSGFMYQGKIPIAGMVVNR

4. LTKSGPSSG

5. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER

6. TLGNVIFMSQVVMQHGACEEKEERYFLLFSSVLIMLSASPR

7. EERYFLLFSSVLIMLSASPRMSGFMYQGK

8. YFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR

9. MSGFMYQGKIPIAGMVVNRLDEIEGSDCMFEITGSTVER

10. IVVHCNNNQDFQEWMEQLNRLTKSGPSSG

Figure 4.6: Peptides that contain at least one non-terminal lysine residue (subset of Figure 4.5).

4.2.5 Cross-linking and peptide pairing

Chemical cross-linking bridges two peptides. To enumerate the di-peptides that may be observed, all the possible pairs of peptides which contain a lysine that could be cross-linked needed to be considered. With the exception of peptide 10 (C-terminal in the protein), all of the peptides in Figure 4.6 were produced by proteolysis at its C-terminal residue. Because a bound cross-linker arm obstructs the proteolytic cleavage of bonds adjacent to the cross-linked residue, C-terminal lysine residues in a peptide could not be considered as a valid site for cross-linking, and so were excluded here. Only peptide pairs where both strands contained a non-terminal lysine were considered. Peptide pairs containing lysines that were too distant in the original structure (see Table 4.2) to cross-link, were excluded. In Table 4.3 below, the range of possible peptide pairs is shown, and 24 distinct cross-linkable peptide pairs are identified. These 24 peptide pairs are listed separately in Figure 4.7 below.

K126	K126	K73	K44	K44	K23 K44	K126	K73	K44	K23		
LTKSG PSSG	IVVHCNNN QDFQEWM EQLNRLTK	MSGFMYQG KIPIAGMVV NR	YFLLFSSVLI MLSASPRM SGFMYQGK	TLGNVIFMS QVVMQHGA CEEKEER	GSSGSSGQ ILSEPIQAW EGDDIKTLG NVIFMSQVV MQHGACEE K	LTK	MSGFMYQG K	TLGNVIFMS QVVMQHGA CEEK	GSSGSSGQ ILSEPIQAW EGDDIK		
									Same peptide	GSSGSSGQILSEPIQAWEGDDIK	K23
								Same peptide	Lysines out of reach	TLGNVIFMSQVVMQHGACEEK	K44
							Same peptide	Lysines out of reach	C-terminal lysine	MSGFMYQGK	K73
						Same peptide	Lysines out of reach	C-terminal lysine	C-terminal lysine	LTK	K126
					Same peptide	C-terminal lysine	C-terminal lysine	K23 & K44 out of reach	K23 & K44 out of reach	GSSGSSGQILSEPIQAWEGDDIKTLG NVIFMSQVVMQHGACEEK	K23 + K44
				Same peptide	K23 & K44 out of reach	C-terminal lysine	C-terminal lysine	Same lysine	Lysines out of reach	TLGNVIFMSQVVMQHGACEEKEER	K44
			Same peptide	Same lysine	K23 & K44 out of reach	C-terminal lysine	C-terminal lysine	Same lysine	Lysines out of reach	YFLLFSSVLIMLSASPRMSGFMYQGK	K44
		Same peptide	C-terminal lysine	19	16	Lysines out of reach	Same lysine	C-terminal lysine	C-terminal lysine	MSGFMYQGKIPIAGMVVNR	K73
	Same peptide	Lysines out of reach	C-terminal lysine	C-terminal lysine	C-terminal lysine	Same lysine	Lysines out of reach	C-terminal lysine	C-terminal lysine	IVVHCNNNQDFQEWMEQLNRLTK	K126
Same peptide	Same lysine	Lysines out of reach	C-terminal lysine	21	18	Same lysine	Lysines out of reach	C-terminal lysine	C-terminal lysine	LTKSGPSSG	K126

Table 4.3: All possible pairs of lysine-containing peptides are tabulated. A number is assigned for peptide pairs that can cross-link to form a di-peptide.

1. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEK + MSGFMYQGKIPIAGMVVNR

2. TLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR

3. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEK + LTKSGPSSG

4. MSGFMYQGKIPIAGMVVNR + LTKSGPSSG

5. MSGFMYQGKIPIAGMVVNR + GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER

6. LTKSGPSSG + GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER

7. MSGFMYQGKIPIAGMVVNR + TLGNVIFMSQVVMQHGACEEKEERYFLLFSSVLIMLSASPR

8. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEK + EERYFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR

9. TLGNVIFMSQVVMQHGACEEKEER + EERYFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR

10. LTKSGPSSG + EERYFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR

11. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + EERYFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR

12. TLGNVIFMSQVVMQHGACEEKEERYFLLFSSVLIMLSASPR + EERYFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR

13. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEK + YFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR

14. TLGNVIFMSQVVMQHGACEEKEER + YFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR

15. LTKSGPSSG + YFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR

16. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + YFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR

17. TLGNVIFMSQVVMQHGACEEKEERYFLLFSSVLIMLSASPR + YFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR

18. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEK + MSGFMYQGKIPIAGMVVNRLDEIEGSDCMFEITGSTVER

19. TLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNRLDEIEGSDCMFEITGSTVER

20. LTKSGPSSG + MSGFMYQGKIPIAGMVVNRLDEIEGSDCMFEITGSTVER

21. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNRLDEIEGSDCMFEITGSTVER

22. TLGNVIFMSQVVMQHGACEEKEERYFLLFSSVLIMLSASPR + MSGFMYQGKIPIAGMVVNRLDEIEGSDCMFEITGSTVER

23. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEK + IVVHCNNNQDFQEWMEQLNRLTKSGPSSG

24. MSGFMYQGKIPIAGMVVNR + IVVHCNNNQDFQEWMEQLNRLTKSGPSSG

25. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + IVVHCNNNQDFQEWMEQLNRLTKSGPSSG

26. EERYFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR + IVVHCNNNQDFQEWMEQLNRLTKSGPSSG

27. YFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR + IVVHCNNNQDFQEWMEQLNRLTKSGPSSG

28. MSGFMYQGKIPIAGMVVNRLDEIEGSDCMFEITGSTVER + IVVHCNNNQDFQEWMEQLNRLTKSGPSSG

Figure 4.7: Peptides that can form di-peptides through cross-linking with BS^3.

4.2.6 Construction of the MS^1 spectrum

4.2.6.1 Calculating the mass of peptides

The mono-isotopic masses of amino acid residues in a peptide are listed in Table 4.4 below. Only the amino acid at the N-terminal of the peptide has an additional hydrogen group and only the amino acid at the C-terminal of the peptide has an additional hydroxyl group, since each of these is only bonded to one adjacent amino acid. Therefore, the mass of a peptide is calculated by summing the masses of each residue in its bonded form and adding the mass of the hydrogen and hydroxyl groups at the peptide termini. The bonded mass of each amino acid is calculated from the chemical formula using empirically-derived atomic masses (Wieser and Coplen, 2011).

	Amino acid symbol	Chemical formula (bonded)	Mono-isotopic Mass (Da)
Hydrogen		H	1.007825
Oxygen		O	15.994915
Hydroxyl		OH	17.002740
Carbon		C	12.000000
Nitrogen		N	14.003074
Sulfur		S	31.972071
Alanine	A	C_3H_5NO	71.0371
Cysteine	C	C_3H_5NOS	103.0092
Aspartic acid	D	$C_4H_5NO_3$	115.0269
Glutamic	E	$C_5H_7NO_3$	129.0426
Phenylalanine	F	C_9H_9NO	147.0684
Glycine	G	C_2H_3NO	57.0215
Histidine	H	$C_6H_7N_3O$	137.0589
Isoleucine	I	$C_6H_{11}NO$	113.0841
Lysine	K	$C_6H_{12}N_2O$	128.0950
Leucine	L	$C_6H_{11}NO$	113.0841
Methionine	M	C_5H_9NOS	131.0405
Asparagine	N	$C_4H_6N_2O_2$	114.0429
Proline	P	C_5H_7NO	97.0528
Glutamine	Q	$C_5H_8N_2O_2$	128.0586
Arginine	R	$C_6H_{12}N_4O$	156.1011
Serine	S	$C_3H_5NO_2$	87.0320
Threonine	T	$C_4H_7NO_2$	101.0477
Valine	V	C_5H_9NO	163.0633
Tryptophan	W	$C_{11}H_{10}N_2O$	186.0793
Tyrosine	Y	$C_9H_9NO_2$	99.0633

Table 4.4: The mono-isotopic masses of each standard amino acid (bound within a peptide chain), as well as several atomic and di-atomic chemical groups, are tabulated. Amino acid masses are based on the chemical formulae denoted and established atomic masses. Atomic masses are sourced from empirical studies (Wieser and Coplen, 2011).

For each of the peptides in Figure 4.6 above which contain a non-terminal lysine, the molecular mass is calculated below.

1. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEK, 4662.2044 Da

2. TLGNVIFMSQVVMQHGACEEKEER, 2734.2879 Da

3. MSGFMYQGKIPIAGMVVNR, 2098.0528 Da

4. LTKSGPSSG, 832.4292 Da

5. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER, 5076.3907 Da

6. TLGNVIFMSQVVMQHGACEEKEERYFLLFSSVLIMLSASPR, 4659.3364 Da

7. EERYFLLFSSVLIMLSASPRMSGFMYQGK, 3386.6867 Da

8. YFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR, 4023.1013 Da

9. MSGFMYQGKIPIAGMVVNRLDEIEGSDCMFEITGSTVER, 4310.0193 Da

10. IVVHCNNNQDFQEWMEQLNRLTKSGPSSG, 3330.5511 Da

4.2.6.2 Calculating the mass of chemically cross-linked di-peptides

The commonly used lysine-reactive cross-linking reagent BS^3 is used in this Chapter for creation of the theoretical di-peptide dataset. Figure 4.8 shows the chemical structure of uncross-linked BS^3. From the chemical structure of BS^3 its chemical formula can be compiled by manual inspection: $C_{16}H_{20}N_2O_{14}S_2$. Based on this chemical formula and the atomic masses listed in Table 4.4, the mass of uncross-linked BS^3 is 528.0356 Da. However, during the cross-linking reaction, BS^3 loses the chemical group $C_4H_4NO_6S$ at each end, as shown in Figure 4.9. Each amino acid residue which is cross-linked also loses a hydrogen atom.

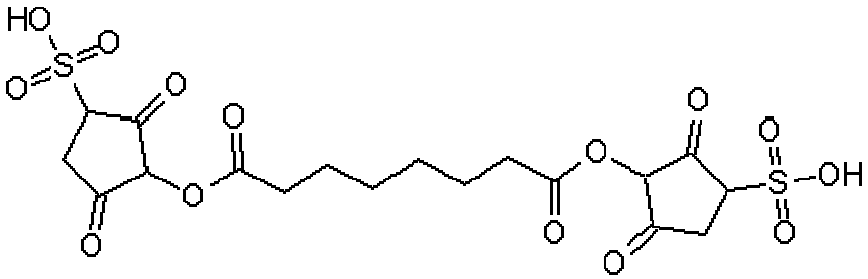


Figure 4.8: The chemical structure of the cross-linking reagent bis(sulfosuccinimidyl) suberate (BS^3).

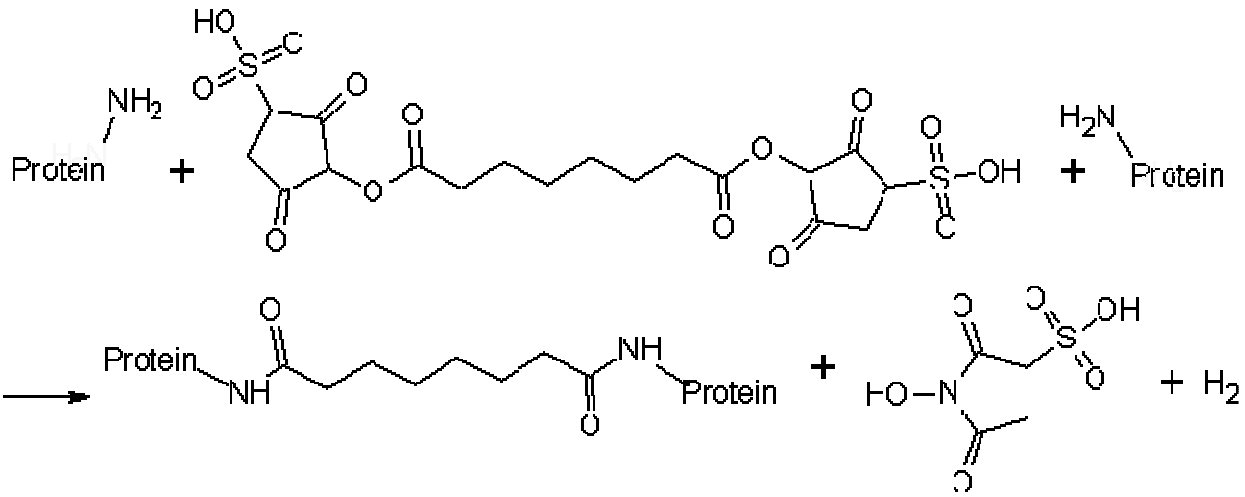


Figure 4.9: The reaction of a generalized NHS-ester (such as BS^3) in the cross-linking of a protein or peptide residue (Kalkhof and Sinz, 2008).

Therefore, the mass that is added to a peptide pair when they are successfully cross-linked by BS^3 can be calculated as follows:

Uncross-linked BS^3	$(C_{16}H_{20}N_2O_{14}S_2)$	528.0356 Da
Chemical groups lost by BS^3	$(C_4H_4NO_6S)_2$	- 387.9519 Da
Chemical groups lost by linked lysine	$(H)_2$	- 2.0157 Da
[Leaving linker arm bonded:	$CO(CH2)_6CO$]	
Mass added to di-peptide by BS^3 linker arm		**<u>138.0681 Da</u>**

The mass of each di-peptide listed above was calculated based on the mass of each peptide as well as the cross-link connecting the two. The mass of the di-peptide that would form from each of the peptide pairs in Figure 4.6 was calculated in this way (see Table 4.5).

Peptide Number[1]	Monoisotopic mass of cross-linked di-peptide (Da)
1	6898.32521
2	4970.40871
3	5632.70161
4	3068.55001
5	7312.51151
6	6046.88791
7	6895.45721
8	9237.56001
9	7309.64351
10	5407.78481
11	9651.74631
12	9234.69201
13	8823.37371
14	6895.45721
15	4993.59851
16	9237.56001
17	8820.50571
18	9110.29171
19	7182.37521
20	5280.51651
21	9524.47801
22	9107.42371
23	8130.82351
24	5566.67191
25	8545.00981
26	7905.90671
27	7491.72041
28	7778.63841

[1.] Numbering from Figure 4.7

Table 4.5: Mono-isotopic masses of BS3 cross-linked di-peptides

4.2.6.3 Calculating the MS[1] peak values for ionized di-peptides

Only ionized di-peptide species can be detected by the mass spectrometer and appear in the MS^1 spectrum. One method of peptide and di-peptide ionization for MS analysis is electro-spray ionization (ESI), where a slightly acidic analyte solution is nebulised in the presence of a strong electric field (reviewed in (Grandori *et al.*, 2009). A peptide or di-peptide becomes ionized through the addition of one or more hydrogen ions. For our purposes here, we will consider only singly charged di-peptide ions where only one hydrogen ion is added.

The peak value which is observed in the mass spectrum depends on the ratio of mass to charge in an ion. The MS^1 spectrum peak values for each di-peptide can be calculated as follows:

Equation 4.2:

$$Peak\ value = \frac{MW_{di-peptide} + MW_{hydrogen}}{charge} = \frac{MW_{di-peptide} + 1.007825Da}{1} = MW_{di-peptide} + 1.007825Da$$

A mass of 1.007825 Da was added to the mass of each of the di-peptides considered, to calculate the peak value of each. Collectively, these peak values constitute an MS^1 spectrum for cross-linked di-peptides, as displayed in Figure 4.10 alongside peaks for single-stranded peptides (from Figure 4.5), in order of ascending value.

361.24519	2361.12119	4660.34419	5779.70769	8131.831335*
433.20469	2517.14029	4663.21219	6047.895735*	8546.017635*
491.21019	2735.29569	4729.10679	6896.465035*	8821.513535*
833.43699	2859.36709	4971.416535*	6896.465035*	8824.381535*
1048.45969*	2973.50819	4994.606335*	6899.333035*	9108.431535*
1069.61929*	3069.557835*	5071.33359	7183.383035*	9111.299535*
1944.06689*	3281.58579	5077.39849	7310.651335*	9235.699835*
2099.06059*	3331.55889	5281.524335*	7313.519335*	9238.567835*
2230.98489*	3387.69449	5408.792635*	7492.728235*	9238.567835*
2321.10939*	4024.10909	5567.679735*	7779.646235*	9525.485835*
2358.25319	4311.02709	5633.709435*	7906.914535*	9652.754135

* Asterix denotes a di-peptide MS^1 peak

Figure 4.10: Single-charge di-peptide peaks in the MS^1 spectrum (including single-stranded peptides).

4.2.6.4 Fragmentation of selected precursors

In order to further characterize precursor ions that were observed in the MS^1 spectrum, precursor ions could be fragmented by collision-induced dissociation (CID). The resultant MS^2 spectrum of fragments is generally distinctive for each precursor ion.

Each peptide bond, in each of the peptides, may fragment during CID. Fragmentation at any given bond creates two fragments: that on the N-terminal and C-terminal sides of the fragmented bond. When charged, these two fragments are referred to as *b* and *y* ions respectively.

Where the fragment contains the cross-linked residue on that peptide, the linker arm and the other peptide are added to the fragment.

We selected a single di-peptide from Figure 4.7, di-peptide number 5, for which we calculated the theoretical fragmentation or MS^2 spectrum. Di-peptide number 5 is, in fact,

two di-peptide precursors, since the longer strand has two cross-linkable lysine residues (K23 and K44). The di-peptide where K23 is cross-linked is designated here as precursor A and the di-peptide where K44 is cross-linked is designated here as precursor B.

GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER +
MSGFMYQGKIPIAGMVVNR

Figure 4.11: The sequence of a di-peptide precursor, selected from Figure 4.7 (di-peptide 5). There are two potential cross-linking sites on the longer strand (β strand), at K23 and K44, and one cross-linking site on the shorter strand (α strand), at K73.

We determined which fragment sequences were possible, considering the fragmentation of each bond. The fragmentation event at each bond results in two fragments. Either of these two fragments may retain a charge and be detected in the MS^2 spectrum. The fragments arising from the 54 possible fragmentation events are shown in Figure 4.12 below. Fragments from the N-terminal side of the fragmented bond (*b* ion fragments) are placed on the left while fragments from the C-terminal side of the fragmented bond (*y* ion fragments) are placed on the right. Where the fragments differ between precursor A and precursor B, the alternative fragments for each are shown, labelled “A” and “B”.

Fragmentation events on longer strand (α strand):

1. G SSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
2. GS SGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
3. GSS GSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
4. GSSG SSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
5. GSSGS SGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
6. GSSGSS GQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
7. GSSGSSG QILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
8. GSSGSSGQ ILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
9. GSSGSSGQI LSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
10. GSSGSSGQIL SEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
11. GSSGSSGQILS EPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
12. GSSGSSGQILSE PIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
13. GSSGSSGQILSEP IQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
14. GSSGSSGQILSEPI QAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
15. GSSGSSGQILSEPIQ AWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
16. GSSGSSGQILSEPIQA WEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
17. GSSGSSGQILSEPIQAW EGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
18. GSSGSSGQILSEPIQAWE GDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
19. GSSGSSGQILSEPIQAWEG DDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
20. GSSGSSGQILSEPIQAWEGD DIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
21. GSSGSSGQILSEPIQAWEGDD IKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
22. GSSGSSGQILSEPIQAWEGDDI KTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
23.
    A. GSSGSSGQILSEPIQAWEGDDIK + MSGFMYQGKIPIAGMVVNR TLGNVIFMSQVVMQHGACEEKEER
    B. GSSGSSGQILSEPIQAWEGDDIK TLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
24.
    A. GSSGSSGQILSEPIQAWEGDDIKT + MSGFMYQGKIPIAGMVVNR LGNVIFMSQVVMQHGACEEKEER
    B. GSSGSSGQILSEPIQAWEGDDIKT LGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
25.
    A. GSSGSSGQILSEPIQAWEGDDIKTL + MSGFMYQGKIPIAGMVVNR GNVIFMSQVVMQHGACEEKEER
    B. GSSGSSGQILSEPIQAWEGDDIKTL GNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
26.
    A. GSSGSSGQILSEPIQAWEGDDIKTLG + MSGFMYQGKIPIAGMVVNR NVIFMSQVVMQHGACEEKEER
    B. GSSGSSGQILSEPIQAWEGDDIKTLG NVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
27.
    A. GSSGSSGQILSEPIQAWEGDDIKTLGN + MSGFMYQGKIPIAGMVVNR VIFMSQVVMQHGACEEKEER
    B. GSSGSSGQILSEPIQAWEGDDIKTLGN VIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
28.
    A. GSSGSSGQILSEPIQAWEGDDIKTLGNV + MSGFMYQGKIPIAGMVVNR IFMSQVVMQHGACEEKEER
    B. GSSGSSGQILSEPIQAWEGDDIKTLGNV IFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
29.
    A. GSSGSSGQILSEPIQAWEGDDIKTLGNVI + MSGFMYQGKIPIAGMVVNR FMSQVVMQHGACEEKEER
    B. GSSGSSGQILSEPIQAWEGDDIKTLGNVI FMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
30.
    A. GSSGSSGQILSEPIQAWEGDDIKTLGNVIF + MSGFMYQGKIPIAGMVVNR MSQVVMQHGACEEKEER
    B. GSSGSSGQILSEPIQAWEGDDIKTLGNVIF MSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
31.
    A. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFM + MSGFMYQGKIPIAGMVVNR SQVVMQHGACEEKEER
    B. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFM SQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
32.
    A. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMS + MSGFMYQGKIPIAGMVVNR QVVMQHGACEEKEER
    B. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMS QVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
33.
    A. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQ + MSGFMYQGKIPIAGMVVNR VVMQHGACEEKEER
    B. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQ VVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR

34.
A. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQV + MSGFMYQGKIPIAGMVVNR VMQHGACEEKEER
B. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQV VMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
35.
A. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVV + MSGFMYQGKIPIAGMVVNR MQHGACEEKEER
B. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVV MQHGACEEKEER + MSGFMYQGKIPIAGMVVNR
36.
A. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVM + MSGFMYQGKIPIAGMVVNR QHGACEEKEER
B. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVM QHGACEEKEER + MSGFMYQGKIPIAGMVVNR
37.
A. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQ + MSGFMYQGKIPIAGMVVNR HGACEEKEER
B. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQ HGACEEKEER + MSGFMYQGKIPIAGMVVNR
38.
A. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQH + MSGFMYQGKIPIAGMVVNR GACEEKEER
B. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQH GACEEKEER + MSGFMYQGKIPIAGMVVNR
39.
A. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHG + MSGFMYQGKIPIAGMVVNR ACEEKEER
B. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHG ACEEKEER + MSGFMYQGKIPIAGMVVNR
40.
A. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGA + MSGFMYQGKIPIAGMVVNR CEEKEER
B. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGA CEEKEER + MSGFMYQGKIPIAGMVVNR
41.
A. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGAC + MSGFMYQGKIPIAGMVVNR EEKEER
B. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGAC EEKEER + MSGFMYQGKIPIAGMVVNR
42.
A. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACE + MSGFMYQGKIPIAGMVVNR EKEER
B. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACE EKE0ER + MSGFMYQGKIPIAGMVVNR
43.
A. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEE + MSGFMYQGKIPIAGMVVNR KEER
B. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEE KEER + MSGFMYQGKIPIAGMVVNR
44. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEK + MSGFMYQGKIPIAGMVVNR EER
45. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKE + MSGFMYQGKIPIAGMVVNR ER
46. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEE + MSGFMYQGKIPIAGMVVNR R

Fragmentation events on shorter strand (β strand):

47. M GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + SGFMYQGKIPIAGMVVNR
48. MS GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + GFMYQGKIPIAGMVVNR
49. MSG GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + FMYQGKIPIAGMVVNR
50. MSGF GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MYQGKIPIAGMVVNR
51. MSGFM GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + YQGKIPIAGMVVNR
52. MSGFMY GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + QGKIPIAGMVVNR
53. MSGFMYQ GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + GKIPIAGMVVNR
54. MSGFMYQG GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + KIPIAGMVVNR
55. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGK IPIAGMVVNR
56. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKI PIAGMVVNR
57. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIP IAGMVVNR
58. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPI AGMVVNR
59. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIA GMVVNR
60. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAG MVVNR
61. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGM VVNR
62. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMV VNR
63. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVV NR
64. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVN R

Figure 4.12: Fragment species resulting from the CID fragmentation of the di-peptides in Figure 4.7.

4.2.6.5 Construction of the MS^2 spectrum for selected precursors

The mass of each of the fragments in Figure 4.12 was calculated by a method similar to that used for the MS^1 spectrum construction in sections 4.2.6.1 and 4.2.6.2. The bonded masses of all the amino acid residues were summed and added to the mass of the chemical groups at the termini of each fragment strand. As with unfragmented peptides, the hydrogen (H^+) and hydroxyl (OH^-) groups were respectively added to the N-terminals and C-terminals which were not adjacent to the fragmented bond. On the N-terminal of the *y* ion fragment, adjacent to the fragmented bond, two hydrogen atoms (H_2) were added as a result of the fragmentation reaction. Where the fragment contained a cross-linked residue, the mass of the BS^3 linker arm as well as the mass of the other, unfragmented peptide was added to the fragment mass.

A

59.037125	762.350825	1872.885525	5524.718325	6607.182125
133.056125	820.392725	1987.912425	5611.710825	6622.163225
146.069125	846.340425	1991.886425	5652.776925	6649.265525
176.127325	860.490225	2102.939325	5751.845325	6679.184725
176.127325	875.434925	2138.954825	5797.790125	6736.224725
220.088125	903.361925	2216.023425	5850.913725	6743.313725
233.101125	923.401925	2252.038925	5868.827225	6777.324125
277.109625	957.543025	2351.107325	5981.954225	6810.225225
290.122625	962.466925	2465.150225	5996.885825	6834.345625
290.170225	994.439025	2522.171725	6109.969925	6864.319725
305.169925	1051.460525	2635.255825	6110.012825	6874.354225
377.154625	1070.627125	2736.303525	6207.022725	6909.293625
389.238625	1091.509525	4562.228725	6227.905125	6921.377625
424.178025	1188.519425	4663.276425	6247.071725	6993.362325
434.212525	1188.562325	4776.360525	6304.093225	7008.362025
464.186625	1301.646425	4833.382025	6336.065325	7008.409625
488.307025	1316.578025	4947.424925	6340.989225	7021.422625
521.208125	1429.705025	5046.493325	6375.130325	7065.431125
555.218525	1447.618525	5082.508825	6395.170325	7078.444125
562.307525	1500.742125	5159.577425	6423.097325	7122.404925
619.347525	1546.686925	5195.592925	6438.042025	7122.404925
649.266725	1645.755325	5306.645825	6452.191825	7152.463125
676.369025	1686.821425	5310.619825	6478.139525	7165.476125
691.350125	1773.813925	5425.646725	6536.181425	7239.495125
718.281825	1815.864025	5437.686325	6551.126125	
747.406125	1860.845925	5482.668225	6580.250425	

B

59.037125	860.490225	3038.503025	4470.149225	6551.126125
133.056125	875.434925	3088.535525	4518.114425	6580.250425
146.069125	903.361925	3141.512225	4569.217625	6622.163225
176.127325	957.543025	3212.549325	4683.260525	6649.265525
176.127325	962.466925	3219.576025	4740.282025	6679.184725
220.088125	1070.627125	3269.570825	4853.366125	6743.313725
233.101125	1091.509525	3306.608025	4954.413825	6777.324125
277.109625	1188.562325	3406.629725	5082.508825	6810.225225
290.122625	1301.646425	3434.666625	5195.592925	6834.345625
290.170225	1429.705025	3533.735025	5310.619825	6864.319725
305.169925	1500.742125	3534.688325	5425.646725	6874.354225
377.154625	1686.821425	3632.803425	5482.668225	6909.293625
389.238625	1815.864025	3665.728825	5611.710825	6921.377625
424.178025	1872.885525	3763.843925	5797.790125	6993.362325
434.212525	1987.912425	3764.797225	5868.827225	7008.362025
464.186625	2102.939325	3863.865625	5996.885825	7008.409625
488.307025	2216.023425	3891.902525	6109.969925	7021.422625
521.208125	2344.118425	3991.924225	6207.022725	7065.431125
555.218525	2445.166125	4028.961425	6227.905125	7078.444125
619.347525	2558.250225	4078.956225	6336.065325	7122.404925
649.266725	2615.271725	4085.982925	6340.989225	7122.404925
676.369025	2729.314625	4157.020025	6395.170325	7152.463125
718.281825	2780.417825	4209.996725	6423.097325	7165.476125
747.406125	2828.383025	4260.029225	6438.042025	7239.495125
762.350825	2909.460425	4357.065125	6452.191825	
846.340425	2941.467125	4389.071825	6536.181425	

Figure 4.13: Fragment peak values in the fragmentation spectra of precursor A and precursor B, of di-peptide 5 in Figure 4.7, are listed in order of ascending value.

4.2.6.6 AnchorMS analysis of constructed dataset

The MS^1 mass spectrum is described by Figure 4.10, and the MS^2 mass spectrum is described by Figure 4.13 above. The MS^1 and MS^2 spectra data, as well as the protein sequence for the PH domain of mouse α-PIX, were uploaded to AnchorMS. AnchorMS was used to calculate MS^1 and MS^2 spectra, using trypsin and BS^3 in the upstream treatment of the sample.

4.3 Results

4.3.1 MS^1 analysis using AnchorMS

The constructed MS^1 spectrum was searched by AnchorMS for possible di-peptide precursors. Figure 4.14 below shows the theoretical MS^1 spectrum of predicted di-peptide precursor peaks, generated by AnchorMS for comparison against the constructed MS^1 spectrum, to flag di-peptide MS^1 peaks. Twenty eight di-peptide precursors were detected by AnchoMS as putative di-peptide precursors, and are enumerated here in Figure 4.15 below. AnchorMS managed to complete a thorough analysis and identified all di-peptides present in the MS^1 spectrum. All of the di-peptide precursors where detected. However, sequence-identical precursors with different cross-linked residues – such as precursors A and B of di-peptide 6 – were not discriminated, but detected as a single peak. This is to be expected since such sequence-identical precursors have the same precursor mass.

1048.45969	4971.416535	6896.465035	7779.646235	9111.299535
1069.61929	4994.606335	6896.465035	7906.914535	9235.699835
1944.06689	5281.524335	6899.333035	8131.831335	9238.567835
2099.06059	5408.792635	7183.383035	8546.017635	9238.567835
2230.98489	5567.679735	7310.651335	8821.513535	9525.485835
2321.10939	5633.709435	7313.519335	8824.381535	
3069.557835	6047.895735	7492.728235	9108.431535	

Figure 4.14: Theoretical MS^1 spectrum generated by AnchorMS for comparison against the uploaded MS^1 spectrum.

1. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEK + MSGFMYQGKIPIAGMVVNR

2. TLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNR

3. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEK + LTKSGPSSG

4. MSGFMYQGKIPIAGMVVNR + LTKSGPSSG

5. MSGFMYQGKIPIAGMVVNR + GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER

6. LTKSGPSSG + GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER

7. MSGFMYQGKIPIAGMVVNR + TLGNVIFMSQVVMQHGACEEKEERYFLLFSSVLIMLSASPR

8. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEK + EERYFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR

9. TLGNVIFMSQVVMQHGACEEKEER + EERYFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR

10. LTKSGPSSG + EERYFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR

11. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + EERYFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR

12. TLGNVIFMSQVVMQHGACEEKEERYFLLFSSVLIMLSASPR + EERYFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR

13. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEK + YFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR

14. TLGNVIFMSQVVMQHGACEEKEER + YFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR

15. LTKSGPSSG + YFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR

16. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + YFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR

17. TLGNVIFMSQVVMQHGACEEKEERYFLLFSSVLIMLSASPR + YFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR

18. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEK + MSGFMYQGKIPIAGMVVNRLDEIEGSDCMFEITGSTVER

19. TLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNRLDEIEGSDCMFEITGSTVER

20. LTKSGPSSG + MSGFMYQGKIPIAGMVVNRLDEIEGSDCMFEITGSTVER

21. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + MSGFMYQGKIPIAGMVVNRLDEIEGSDCMFEITGSTVER

22. TLGNVIFMSQVVMQHGACEEKEERYFLLFSSVLIMLSASPR + MSGFMYQGKIPIAGMVVNRLDEIEGSDCMFEITGSTVER

23. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEK + IVVHCNNNQDFQEWMEQLNRLTKSGPSSG

24. MSGFMYQGKIPIAGMVVNR + IVVHCNNNQDFQEWMEQLNRLTKSGPSSG

25. GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER + IVVHCNNNQDFQEWMEQLNRLTKSGPSSG

26. EERYFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR + IVVHCNNNQDFQEWMEQLNRLTKSGPSSG

27. YFLLFSSVLIMLSASPRMSGFMYQGKIPIAGMVVNR + IVVHCNNNQDFQEWMEQLNRLTKSGPSSG

28. MSGFMYQGKIPIAGMVVNRLDEIEGSDCMFEITGSTVER + IVVHCNNNQDFQEWMEQLNRLTKSGPSSG

Figure 4.15: Di-peptide species detected by AnchorMS in the uploaded MS^1 *m/z* list.

4.3.2 MS^2 analysis using AnchorMS

4.3.2.1 AnchorMS identifies the correct di-peptide precursor over a sequence-identical alternative candidate precursor

To investigate the ability of AnchorMS to assign the correct di-peptide precursor to an observed MS^2 spectrum, an AnchorMS analysis was performed on the constructed MS^2 spectrum. The constructed MS^2 fragment spectra for di-peptide precursor A (see Figure 4.13 above) was uploaded to AnchorMS and analysed to identify the di-peptide precursor, as well as to ascertain which of its residues were cross-linked. AnchorMS matched the observed spectrum against the generated theoretical spectra (see Figure 4.16 below) for precursors cross-linked at different sites (precursors A and B), under various matching tolerances.

Under all tolerance values, a set of 90 peaks with identical *m/z* values were matched. These represent the shared fragments present in both MS^2 spectra. Table 4.6 below describes the false matches that occurred for tolerance values less than 1000 ppm.

At the moderate-stringency tolerance of 50 ppm, the theoretical precursor candidates A and B (cross-linked at K23 and cross-linked at K44, respectively) matched 128 and 92 fragment peaks, respectively. Both scores are well above the calculated decoy matching baseline of 6.174, suggesting that at least one of these precursor candidates is a true assignment. From the separation in *N* score values, precursor A presents as the confident and true assignment (*N* score is 36 or 39.13% higher than precursor B). Because the

precursor candidates are sequence-identical, AnchorMS calculated and displayed the *UMC* score in addition to the *N* score and estimated *D* score. Unique fragments are highlighted in the theoretical spectra shown in Figure 4.16.

The candidate precursors A scored an *UMC* of 38 while candidate precursor B scored a definitive *UMC* of 0. Although there are other peaks within the tolerance of the shared fragment peaks from fragmentation events 4 and 63 (see rows 1 and 6 in Table 4.6), the correct peaks were closer in *m/z* value and during MS^2 matching, AnchorMS only included the closest matches in the *N* and *UMC* scores. The *UMC* scores unequivocally indicate precursor A as the correct assignment (*UMC* score is 38 higher than precursor B).

The high separation in both *N* and *UMC* scores was, in part, due to the large number of residues separating the alternative cross-linking sites (20 residues), granting each candidate precursor a substantial number of unique fragments to distinguish its MS^2 spectrum from the other. Under these conditions, even if only a fraction of the unique fragment peaks were observed in the experimental spectrum, there would be sufficient remaining unique fragment peaks to distinguish the correct assignment.

AnchorMS identified the correct peptide sequences of the di-peptide precursor and managed to correctly ascertain which residues were cross-linked. The results of the analysis are shown in Table 4.7 below.

	Fragment in precursor A spectrum[1]	**Fragment in precursor B spectrum[1]**	**Peak in MS^2 spectrum of precursor A**	**Peak in MS^2 spectrum of precursor B**	**Minimum absolute tolerance for match (Da)**	**Minimum relative tolerance for match (ppm)**
1	*b*-ion of 4	*y*-ion of 63	290.1226	290.1226	0.0476	164.0686
2	*y*-ion of 37A	*b*-ion of 13	1188.5194	1188.5623	0.0429	36.0953
3	*b*-ion of 30A	*y*-ion of 20	5306.6458	5310.6198	3.9740	748.8723
4	*b*-ion of 37A	*y*-ion of 13	6110.0128	6109.9699	0.0429	7.0213
5	*y*-ion of 11	*b*-ion of 56	6336.0653	6340.9892	4.9239	777.1227
6	*b*-ion of 63	*y*-ion of 4	7008.3620	7008.4096	0.0476	6.7919
[1.] Numbering from Figure 4.12						

Table 4.6: False positive MS^2 peak matches which occurred between the spectra of precursor A and precursor B under various minimum tolerances.

A

59.037125	762.350825	1872.885525	5524.718325	6607.182125
133.056125	820.392725	1987.912425	5611.710825	6622.163225
146.069125	846.340425	1991.886425	5652.776925	6649.265525
176.127325	860.490225	2102.939325	5751.845325	6679.184725
176.127325	875.434925	2138.954825	5797.790125	6736.224725
220.088125	903.361925	2216.023425	5850.913725	6743.313725
233.101125	923.401925	2252.038925	5868.827225	6777.324125
277.109625	957.543025	2351.107325	5981.954225	6810.225225
290.122625	962.466925	2465.150225	5996.885825	6834.345625
290.170225	994.439025	2522.171725	6109.969925	6864.319725
305.169925	1051.460525	2635.255825	6110.012825	6874.354225
377.154625	1070.627125	2736.303525	6207.022725	6909.293625
389.238625	1091.509525	4562.228725	6227.905125	6921.377625
424.178025	1188.519425	4663.276425	6247.071725	6993.362325
434.212525	1188.562325	4776.360525	6304.093225	7008.362025
464.186625	1301.646425	4833.382025	6336.065325	7008.409625
488.307025	1316.578025	4947.424925	6340.989225	7021.422625
521.208125	1429.705025	5046.493325	6375.130325	7065.431125
555.218525	1447.618525	5082.508825	6395.170325	7078.444125
562.307525	1500.742125	5159.577425	6423.097325	7122.404925
619.347525	1546.686925	5195.592925	6438.042025	7122.404925
649.266725	1645.755325	5306.645825	6452.191825	7152.463125
676.369025	1686.821425	5310.619825	6478.139525	7165.476125
691.350125	1773.813925	5425.646725	6536.181425	7239.495125
718.281825	1815.864025	5437.686325	6551.126125	
747.406125	1860.845925	5482.668225	6580.250425	

B

59.037125	860.490225	3038.503025	4470.149225	6551.126125
133.056125	875.434925	3088.535525	4518.114425	6580.250425
146.069125	903.361925	3141.512225	4569.217625	6622.163225
176.127325	957.543025	3212.549325	4683.260525	6649.265525
176.127325	962.466925	3219.576025	4740.282025	6679.184725
220.088125	1070.627125	3269.570825	4853.366125	6743.313725
233.101125	1091.509525	3306.608025	4954.413825	6777.324125
277.109625	1188.562325	3406.629725	5082.508825	6810.225225
290.122625	1301.646425	3434.666625	5195.592925	6834.345625
290.170225	1429.705025	3533.735025	5310.619825	6864.319725
305.169925	1500.742125	3534.688325	5425.646725	6874.354225
377.154625	1686.821425	3632.803425	5482.668225	6909.293625
389.238625	1815.864025	3665.728825	5611.710825	6921.377625
424.178025	1872.885525	3763.843925	5797.790125	6993.362325
434.212525	1987.912425	3764.797225	5868.827225	7008.362025
464.186625	2102.939325	3863.865625	5996.885825	7008.409625
488.307025	2216.023425	3891.902525	6109.969925	7021.422625
521.208125	2344.118425	3991.924225	6207.022725	7065.431125
555.218525	2445.166125	4028.961425	6227.905125	7078.444125
619.347525	2558.250225	4078.956225	6336.065325	7122.404925
649.266725	2615.271725	4085.982925	6340.989225	7122.404925
676.369025	2729.314625	4157.020025	6395.170325	7152.463125
718.281825	2780.417825	4209.996725	6423.097325	7165.476125
747.406125	2828.383025	4260.029225	6438.042025	7239.495125
762.350825	2909.460425	4357.065125	6452.191825	
846.340425	2941.467125	4389.071825	6536.181425	

Figure 4.16: Theoretical MS^2 spectrum generated by AnchorMS for comparison against the uploaded MS^2 spectrum.

	Protein containing α-strand	Protein containing β-strand	α strand	β strand	α-strand cross-link site	β-strand cross-link site	Max C_α -C_α span (Å)	*N* score	*D* score	*UMC* score
1	1V61:A|Pleckstrin Homology domain of mouse alph-PIX	1V61:A|Pleckstrin Homology domain of mouse alph-PIX	GSSGSSGQILSEPIQAWEGDDIKTLGNVIFMSQVVMQHGACEEKEER	MSGFMYQGKIPIAGMVVNR	23	3	24.34	128	6.145	38

Table 4.7: Di-peptide identified by AnchorMS as the precursor of the uploaded MS^2 spectrum.

4.3.2.2 AnchorMS identifies the correct di-peptide precursor over a sequence-shuffled alternative candidate precursor

Having demonstrated AnchorMS' ability to distinguish between sequence-identical di-peptide precursors, we sought to examine AnchorMS' ability discriminate false assignments that differ in sequence. A decoy di-peptide was generated by shuffling the sequence of precursor A. In a similar fashion to the constructed spectra, we generated an MS^2 spectrum for this sequence-shuffled decoy precursor, designated here as precursor C. Because the residue composition of precursor C is identical to that of precursor A, they match at the MS^1 level and so precursor A is the putative MS^1 identity that AnchorMS considers.

GPTIIIQMGKNSAIYGFIMNSFWEEAGVLLEVEMKSQIPQDEESRSG + HMVQSGRGGKASCGVDMQV

Figure 4.17: Decoy di-peptide precursor C, generated by shuffling the sequence of precursor A, across strands, and cross-linked at residue 35 and residue 10 of the longer (α) and shorter (β) strands, respectively.

The MS^2 spectrum of precursor C (see Figure 4.18 below) was uploaded to AnchorMS and analysed to determine its precursor identity and cross-linked residues. In particular, we sought to ascertain whether AnchorMS would confirm or reject the putative precursor assignment of precursor A.

AnchorMS matched the observed spectrum against the generated theoretical spectra for precursor A (see Figure 4.16A above) under a moderate tolerance of 5 ppm. At this tolerance an *N* score of 0 was obtained, well below the calculated decoy matching baseline (*D* score) of 5.97. By increasing the tolerance to 0.045 Da or 6.6 ppm, a single false positive fragment peak was matched (*D* score = 6.00). Even at the low-stringency tolerance of 500 ppm, the number of fragment peak matches (*N* score = 5) is below the estimated false positive baseline (*D* score = 6.84).

Because its *N* score fell below the *D* score false positive baseline, it was correctly determined that the precursor of the uploaded MS^2 spectrum was not precursor A, but rather a false precursor match. AnchorMS thus demonstrated its ability to identify false positive precursor matches, as would be the case where none of the candidate assignments were correct.

59.037125	780.324925	2571.302025	5147.409925	6633.234225
77.047725	797.396625	2700.344625	5261.452825	6649.221725
119.094625	839.351725	2829.387225	5392.493325	6657.236725
139.074525	854.418125	2900.424325	5505.577425	6702.142325
156.089925	855.489025	2957.445825	5652.645825	6706.243225
164.079725	908.383525	3056.514225	5709.667325	6714.258225
247.153225	910.388825	3169.598325	5872.730625	6762.276825
257.137625	911.439625	3282.682425	5964.921225	6801.290225
270.115025	912.510525	3411.725025	5985.814725	6805.311625
320.180825	1005.436325	3510.793425	6051.953225	6815.226425
369.183425	1040.605525	3527.655725	6056.851825	6891.319425
370.221725	1118.520425	3639.836025	6143.883825	6920.338525
378.193725	1154.648425	3658.696225	6180.011825	6928.310525
407.212825	1241.680425	3770.876525	6257.926725	6929.348825
483.305825	1246.579025	3787.738825	6293.095925	6978.351425
493.220625	1312.717525	3886.807225	6386.021725	7028.417225
497.242025	1333.611025	4015.849825	6387.092625	7041.394625
536.255425	1425.801625	4128.933925	6388.143425	7051.379025
584.274025	1588.864925	4242.018025	6390.148725	7134.452525
592.289025	1645.886425	4341.086425	6443.043225	7142.442325
596.389925	1792.954825	4398.107925	6444.114125	7159.457725
641.295525	1906.038925	4469.145025	6459.180525	7179.437625
649.310525	2037.079425	4598.187625	6501.135625	7221.484525
665.298025	2151.122325	4727.230225	6518.207325	7239.495125
724.448525	2238.154325	4913.309525	6546.212525	
752.319725	2385.222725	5060.377925	6574.083725	

Figure 4.18: The generated MS^2 spectrum of decoy precursor C (shuffled sequence of precursor A)

4.4 Closing Discussion

We have described, in detail, the manual construction of a theoretical dataset. We tested AnchorMS using this constructed dataset, as well as with a similarly constructed dataset based on a sequence-shuffled decoy precursor.

We confirmed that AnchorMS functioned as intended with an actual analysis run. The results showed that AnchorMS was able to identify di-peptide species present in the generated data, and that the AnchorMS analysis was consistent with established atomic and amino acid masses, as well as the mechanisms that applied to protease digestion, protein cross-linking and CID fragmentation of peptide species.

In our analysis of the MS^2 spectra of sequence-identical precursors, we demonstrated the ability of AnchorMS to identify the correct precursor from amongst similar alternative candidate assignments, and showed the utility of the *UMC* score in determining the correct cross-linked residues. We also demonstrated the ability of AnchorMS to exclude false positive precursor matches. In both analyses, the *N* scores of incorrect precursor assignments, for reasonable tolerance values, were consistently below the estimated *D* score false positive baseline.

AnchorMS appeared to effectively function as intended. Nonetheless, future refinement of AnchorMS would benefit from testing against more complex datasets, such as those obtained from large protein complexes or against poor quality experimental mass spectra.

4.5 References

1. Berman, H., Henrick, K., and Nakamura, H. (2003) Announcing the worldwide Protein Data Bank. *Nat.Struct.Biol.* **10,** 980.

2. Grandori, R., Santambrogio, C., Brocca, S., Invernizzi, G., and Lotti, M. (2009) Electrospray-ionization mass spectrometry as a tool for fast screening of protein structural properties. *Biotechnol.J.* **4,** 73-87.

3. Kalkhof, S. and Sinz, A. (2008) Chances and pitfalls of chemical cross-linking with amine-reactive N-hydroxysuccinimide esters. *Anal.Bioanal.Chem.* **392,** 305-312.

4. Manser, E., Loo, T. H., Koh, C. G., Zhao, Z. S., Chen, X. Q., Tan, L., Tan, I., Leung, T., and Lim, L. (1998) PAK kinases are directly coupled to the PIX family of nucleotide exchange factors. *Mol.Cell.* **1,** 183-192.

5. Mayer, B. J., Ren, R., Clark, K. L., and Baltimore, D. (1993) A putative modular domain present in diverse signaling proteins. *Cell.* **73,** 629-630.

6. Musacchio, A., Gibson, T., Rice, P., Thompson, J., and Saraste, M. (1993) The PH domain: a common piece in the structural patchwork of signalling proteins. *Trends Biochem.Sci.* **18,** 343-348.

7. Rose, P. W., Bi, C., Bluhm, W. F., Christie, C. H., Dimitropoulos, D., Dutta, S., Green, R. K., Goodsell, D. S., Prlic, A., Quesada, M., Quinn, G. B., Ramos, A. G., Westbrook, J. D., Young, J., Zardecki, C., Berman, H. M., and Bourne, P. E. (2013) The RCSB Protein Data Bank: new resources for research and education. *Nucleic Acids Res.* **41,** D475-D482.

8. Serpa, J. J., Parker, C. E., Petrotchenko, E. V., Han, J., Pan, J., and Borchers, C. H. (2012) Mass spectrometry-based structural proteomics. *Eur.J.Mass Spectrom.(Chichester, Eng).* **18,** 251-267.

9. Siepen, J. A., Keevil, E. J., Knight, D., and Hubbard, S. J. (2007) Prediction of missed cleavage sites in tryptic peptides aids protein identification in proteomics. *J.Proteome.Res.* **6,** 399-408.

10. Wieser, M. E. and Coplen, T. B. (2011) Atomic weights of the elements 2009 (IUPAC

Technical Report). *Pure Appl.Chem.* **83,** 359-396.

11. Yu, J. W., Mendrola, J. M., Audhya, A., Singh, S., Keleti, D., DeWald, D. B., Murray, D., Emr, S. D., and Lemmon, M. A. (2004) Genome-wide analysis of membrane targeting by S. cerevisiae pleckstrin homology domains. *Mol.Cell.* **13,** 677-688.

Chapter 5
Discussion

5.1 Manuscript Overview

MS3D refers to methodologies of structural investigation involving the mass spectrometric analysis of cross-linked proteins or protein complexes, typically post-digestion by proteolytic enzymes. Cross-linked di-peptides identified in the sample indicate the maximum distance between cross-linked residues based on the molecular reach of the cross-linking reagent, which may be pertinent to refinement of the structural model of the protein or protein complex (Mayne and Patterton, 2011).

In this manuscript, the MS3D technique was introduced, along with its role in the field of structural biology and its computational analysis requirements. We have detailed each of the available software packages used to interpret mass spectra for MS3D analysis and summarized their capabilities. Our own AnchorMS software package was developed in Python and presented as an alternative, free web-accessible tool for the identification of cross-linked di-peptides in an MS3D analysis. In response to difficulties in acquiring experimental MS data for di-peptide analytes, a simulation of decoy MS^2 spectra matching was run to statistically determine the likely rate of false positive matching under a variety of conditions. The simulation data was modelled using both standard and custom mathematical functions which were combined and calibrated against the simulation data to form a mathematical model for the estimation of false positive matching levels in terms of precursor size, precursor charge and peak-matching tolerance. With this model implemented as a baseline threshold in the scoring scheme, AnchorMS provides a novel and viable alternative to existing MS3D software. The completed AnchorMS package was tested using a constructed theoretical MS dataset, which demonstrated the functionality of each of the AnchorMS components.

This manuscript has described the derivation of the scoring scheme and the final software package which will remain freely accessible (cbio.ufs.ac.za/AnchorMS), along with all simulation data used in the calibration of the false positive threshold equations (downloadable at cbio.ufs.ac.za/AnchorMS/Development/).

5.2 Changes in MS3D software over time

In chapter 1 we can see that the sophistication of released MS3D software has increased over time. The earliest MS3D packages released before 2003, such as ASAP, MS-Bridge and X-Link, provided no support for tandem MS analysis, nor any form of match scoring (Clauser *et al.*, 1999; Taverner *et al.*, 2002; Young *et al.*, 2000). At the commencement of AnchorMS development in 2007, SearchXLinks, Pro-Crosslink and XLINK had only just been released (Gao *et al.*, 2006; Seebacher *et al.*, 2006; Wefing *et al.*, 2006). Each was specifically geared towards data produced from very specific experimental designs. At this point, there appeared to be a clear need for a general-purpose MS3D tool that did not make any assumptions about the analyte sample. Also, efforts to apply existing methods to the estimation of false positive di-peptide matching had only just begun with the release of iXLINK. In this environment, the development and release of AnchorMS filled clear gaps in the available functionality amongst the bioinformatic tools then available for MS3D analysis. Since the commencement of AnchorMS development in 2007 a large number of MS3D tools have been released and many of them have improved on the methods of prior packages, offering greater sophistication in their analysis. All but a few packages released since, include tandem MS analysis and some form of scoring for spectrum matching. Several programs now apply complex scoring schemes in order to evaluate the likelihood that a spectrum match is a true positive, each implementing distinctive approaches to estimating the probability of a spectrum match being a true positive (McIlwain *et al.*, 2010; Rinner *et al.*, 2008; Xu *et al.*, 2008). The importance of handling large datasets, while not fully supported, has also been acknowledged and addressed (Lee *et al.*, 2007; Rinner *et al.*, 2008).

Even though the development of AnchorMS was conceived and undertaken in 2007, a number of obstacles surfaced in the intervening period to greatly delay the completion and release of AnchorMS. In the face these delays, the majority of the latest software packages were released after the AnchorMS release date originally projected. Consequently, in the current environment, the novelty of AnchorMS is reduced with many alternatives now available to the user. Nonetheless, AnchorMS still offers a number of unique functionalities, as reviewed below.

5.3 AnchorMS fulfils a role as a generally applicable MS3D tool

Even modern MS3D packages tend to be designed around catering to very specific experimental designs or scenarios, such as the use of isotope labelling. AnchorMS addresses this by providing an MS3D analysis tool which is generally applicable and does not depend on specific up-stream sample treatments or control datasets. This allows for a variety of previously generated datasets to be analysed using AnchorMS, irrespective of experimental design.

5.4 AnchorMS is uniquely equipped for the analysis of sequence-identical di-peptides

As with many previously released MS3D tools, AnchorMS uses the number of matches as a primary score measure for distinguishing the best assignment to an observed spectrum (*N* Score). This simple closeness-of-fit measure is generally sufficient to discriminate between alternative candidate assignments. However, the analysis of di-peptides inevitably encounters di-peptides with identical strand sequences but different cross-linked residues. This class of precursors cannot be distinguished at the MS^1 level because the precursor masses are identical (isobaric) and is also difficult to distinguish at the MS^2 level, especially on the basis of an *N* Score alone, since they share many of their fragment peaks. AnchorMS is unique in specifically catering for this class of precursors, by considering the possible co-fragmentation of isobaric di-peptides in MS^2 spectra and employing a unique and specialised secondary score (*UMC* Score) for distinguishing between sequence-identical precursors. These two features render AnchorMS better equipped to interpret MS^2 spectra from di-peptides with uncertain cross-linking sites, and moreover discern if, in fact, a spectrum is due to the co-fragmentation of more than one precursor candidate. The latter is not considered by other MS3D tools, presumably on the assumption that upstream sample chromatography will sufficiently separate sequence-identical di-peptides to avoid co-fragmentation.

5.5 AnchorMS applies a unique mathematical model as a dynamic false positive threshold

MS3D tools such as MS2PRO (Kruppa *et al.*, 2003) and Pro-Crosslink (Gao *et al.*, 2006) simply calculate a score for spectrum matching but do not incorporate a means by which to evaluate whether a score is sufficient to indicate a confident assignment.

This is essential for identifying false matches. AnchorMS addresses this problem by deriving a unique mathematical model for estimating false matching levels, based on simulated false positive matching over a large number of di-peptide sequences. The data set generated by this simulation is unbiased and inclusive, surveying only randomly generated di-peptides, thereby preventing the mathematical model from being biased towards any sub-set of possible di-peptides. The mathematical model also provides AnchorMS with a dynamic threshold which allows AnchorMS to more accurately discriminate false positive matches by adjusting its level according to the tolerance, precursor charge and precursor length, each of which affects the levels of false matching.

5.6 AnchorMS is consistently accessible through a simple web interface

While many MS3D software packages offer useful functionality their links are not always well maintained and may be difficult to obtain or access several years after their initial release. AnchorMS is newly released on a well-maintained server ensuring its functional accessibility for the foreseeable future. This allows AnchorMS to fill the gaps left by MS3D tools with lapsed web links. The AnchorMS web interface is also simple and easy to use with ample annotation through its 'help' links. By releasing AnchorMS primarily as a web service, issues relating to differing versions and platforms are avoided in the future.

5.7 AnchorMS supplies inferred distance constraints for structural modelling but does not include structural modelling functionality

The MS3D workflow involves a number of distinct steps: The protein complex is cross-linked and digested and its MS^1 as well as its MS^2 spectra are acquired, and possibly processed. Mass spectrometric analysis (with or without upstream analyte separation) is conducted to identify cross-linked species and residues. Inter-residue distances are inferred from cross-linked species and, finally, structural models are refined based on these inter-residue distances. AnchorMS performs all the mass spectrometric analysis required to identify the cross-linked peptides and residues in the sample, as well as the inference of the maximum inter-residue distance from each di-peptide detected. Through these functions, AnchorMS forms part of a larger MS3D digital workflow or pipeline of software tools where the output of one tool is accepted as the input of the next tool. The functionalities outside of AnchorMS, however, are not unique to MS3D or

cross-linked species, and are used for a variety of experimental workflows.

Preceding AnchorMS in this workflow, instrument platforms, as well as a range of specialised software packages, process the acquired raw mass spectrometric data files to simplify their down-stream analysis. Such processing can take many forms, including the exclusion of noise (Mo *et al.*, 2010; Morohashi *et al.*, 2007; Wenig and Odermatt, 2010), as well as charge deconvolution to generate a mass list from the peak list and deisotoping to remove the peaks representing low-abundance isotopes (Dobo and Kaltashov, 2001; Jaitly *et al.*, 2009; Pluskal *et al.*, 2010; Zhang and Marshall, 1998). Preprocessing may also involve the conversion of proprietary, raw data file formats into data formats which are more widely accessible, such as the increasingly standard XML-based mzML file format (Deutsch, 2010). AnchorMS incorporates some of this workflow into its design by accepting certain proprietary data formats (.raw, .wiff, .d) and converting them into mzML format using the utility MSConvert, stored and run locally on the AnchorMS server.

Subsequent to AnchorMS in the workflow, a wide variety of structural modelling packages can be utilized for structural refinement. At the most rudimentary level structural models can be manually inspected and evaluated based on the inter-residue distances obtained from MS3D analysis, using programmes such as Friend (Abyzov *et al.*, 2005), VMD (Humphrey *et al.*, 1996), Jmol (HerrÃ¡ez, 2006), Pymol (DeLano, 2002), BioBlender (Andrei *et al.*, 2010) or Polyview-3D (Porollo and Meller, 2007). Some packages are specifically designed to optimize the putative structure based on available data (reviewed in (Gront *et al.*, 2012)), including Protein-Hub (Jo and Tanaka, 2010), CYANA (Guntert, 2004), VMD-XPLOR (Schwieters and Clore, 2001) or XPLOR-NIH (Schwieters *et al.*, 2003). Since MS3D is essentially a low resolution structural tool, it is better suited for excluding erroneous models and adjusting inaccurate structure elements, than for constructing new structural models.

5.8 Summation

MS3D methodologies are increasingly being employed to investigate the structures of large protein complexes. Because it is relatively inexpensive and rapid, the likely importance of MS3D also lies in its ability to increase the rate of structural elucidation (over that using traditional methods alone) in comparison to the rate of gene discovery and annotation, and so contribute to the overall progress of functional genomics. The

bioinformatic analysis underpinning MS3D will prove critical to its contribution as a technique. AnchorMS is released as an alternative tool for MS3D analysis with a distinctive mathematical model that serves as a dynamic false positive scoring threshold and giving special consideration to difficulties associated with sequence-identical di-peptides. AnchorMS is released as a free web service, forming part of a larger digital workflow employed for answering structural questions through MS3D methodologies.

5.9 References

1. Abyzov, A., Errami, M., Leslin, C. M., and Ilyin, V. A. (2005) Friend, an integrated analytical front-end application for bioinformatics. *Bioinformatics.* **21,** 3677-3678.

2. Andrei, R. M., Callieri, M., Zini, M. F., Loni, T., Maraziti, G., Pan, M. C., and ZoppÃ¨, M. (2010) BioBlender: A Software for Intuitive Representation of Surface Properties of Biomolecules. *arXiv preprint arXiv:1009.4674*

3. Clauser, K. R., Baker, P., and Burlingame, A. L. (1999) Role of accurate mass measurement (+/- 10 ppm) in protein identification strategies employing MS or MS/MS and database searching. *Anal.Chem.* **71,** 2871-2882.

4. DeLano, W. L. (2002) The PyMOL Molecular Graphics System (2002) DeLano Scientific, San Carlos, CA, USA. *Computer Program*

5. Deutsch, E. W. (2010) Mass spectrometer output file format mzML. *Methods Mol.Biol.* **604,** 319-331.

6. Dobo, A. and Kaltashov, I. A. (2001) Detection of multiple protein conformational ensembles in solution via deconvolution of charge-state distributions in ESI MS. *Analytical chemistry* **73,** 4763-4773.

7. Gao, Q., Xue, S., Doneanu, C. E., Shaffer, S. A., Goodlett, D. R., and Nelson, S. D. (2006) Pro-CrossLink. Software tool for protein cross-linking and mass spectrometry. *Anal.Chem.* **78,** 2145-2149.

8. Gront, D., Kmiecik, S., Blaszczyk, M., Ekonomiuk, D., and KoliÅ„ski, A. (2012) Optimization of protein models. *Wiley Interdisciplinary Reviews: Computational Molecular Science*

9. Guntert, P. (2004) Automated NMR structure calculation with CYANA. *METHODS IN MOLECULAR BIOLOGY-CLIFTON THEN TOTOWA-* **278,** 353-378.

10. HerrÃ¡ez, A. (2006) Biomolecules in the computer: Jmol to the rescue. *Biochemistry and Molecular Biology Education* **34,** 255-261.

11. Humphrey, W., Dalke, A., and Schulten, K. (1996) VMD: visual molecular dynamics. *Journal of molecular graphics* **14,** 33-38.

12. Jaitly, N., Mayampurath, A., Littlefield, K., Adkins, J. N., Anderson, G. A., and Smith, R. D. (2009) Decon2LS: An open-source software package for automated processing and visualization of high resolution mass spectrometry data. *BMC.Bioinformatics.* **10,** 87.

13. Jo and Tanaka. Protein-hub; Automatic Protein Structure Prediction & Optimization System. (2010) 9th Community Wide Experiment on the Critical Assessment of Techniques for Protein Structure Prediction

14. Kruppa, G. H., Schoeniger, J., and Young, M. M. (2003) A top down approach to protein structural studies using chemical cross-linking and Fourier transform mass spectrometry. *Rapid Commun.Mass Spectrom.* **17,** 155-162.

15. Lee, Y. J., Lackner, L. L., Nunnari, J. M., and Phinney, B. S. (2007) Shotgun cross-linking analysis for studying quaternary and tertiary protein structures. *J.Proteome.Res.* **6,** 3908-3917.

16. Mayne, S. L. and Patterton, H. G. (2011) Bioinformatics tools for the structural elucidation of multi-subunit protein complexes by mass spectrometric analysis of protein-protein cross-links. *Brief.Bioinform.* **12,** 660-671.

17. McIlwain, S., Draghicescu, P., Singh, P., Goodlett, D. R., and Noble, W. S. (2010) Detecting cross-linked peptides by searching against a database of cross-linked peptide pairs. *J.Proteome.Res.* **9,** 2488-2495.

18. Mo, F., Mo, Q., Chen, Y., Goodlett, D. R., Hood, L., Omenn, G. S., Li, S., and Lin, B. (2010) WaveletQuant, an improved quantification software based on wavelet signal threshold de-noising for labeled quantitative proteomic analysis. *BMC.Bioinformatics.* **11,** 219-11.

19. Morohashi, M., Shimizu, K., Ohashi, Y., Abe, J., Mori, H., Tomita, M., and Soga, T. (2007) P-BOSS: a new filtering method for treasure hunting in metabolomics. *J.Chromatogr.A.* **1159,** 142-148.

20. Pluskal, T. Å., Castillo, S., Villar-Briones, A., and OreÅ¡iÄ , M. (2010) MZmine 2: modular framework for processing, visualizing, and analyzing mass spectrometry-based molecular profile data. *BMC Bioinformatics* **11,** 395.

21. Porollo, A. and Meller, J. (2007) Versatile annotation and publication quality visualization of protein complexes using POLYVIEW-3D. *BMC.Bioinformatics.* **8:316.,** 316.

22. Rinner, O., Seebacher, J., Walzthoeni, T., Mueller, L. N., Beck, M., Schmidt, A., Mueller, M., and Aebersold, R. (2008) Identification of cross-linked peptides from large sequence databases. *Nat.Methods* **5,** 315-318.

23. Schwieters, C. D. and Clore, G. M. (2001) The VMD-XPLOR visualization package for NMR structure refinement. *J.Magn Reson.* **149,** 239-244.

24. Schwieters, C. D., Kuszewski, J. J., Tjandra, N., and Marius Clore, G. (2003) The Xplor-NIH NMR molecular structure determination package. *Journal of Magnetic Resonance* **160,** 65-73.

25. Seebacher, J., Mallick, P., Zhang, N., Eddes, J. S., Aebersold, R., and Gelb, M. H. (2006) Protein cross-linking analysis using mass spectrometry, isotope-coded cross-linkers, and integrated computational data processing. *J.Proteome.Res.* **5,** 2270-2282.

26. Taverner, T., Hall, N. E., O'Hair, R. A., and Simpson, R. J. (2002) Characterization of an antagonist interleukin-6 dimer by stable isotope labeling, cross-linking, and mass spectrometry. *J.Biol.Chem.* **277,** 46487-46492.

27. Wefing, S., Schnaible, V., and Hoffmann, D. (2006) SearchXLinks. A program for the identification of disulfide bonds in proteins from mass spectra. *Anal.Chem.* **78,** 1235-1241.

28. Wenig, P. and Odermatt, J. (2010) OpenChrom: a cross-platform open source software for the mass spectrometric analysis of chromatographic data. *BMC Bioinformatics* **11,** 405.

29. Xu, H., Zhang, L., and Freitas, M. A. (2008) Identification and characterization of disulfide bonds in proteins and peptides from tandem MS data by use of the MassMatrix MS/MS search engine. *J.Proteome.Res.* **7,** 138-144.

30. Young, M. M., Tang, N., Hempel, J. C., Oshiro, C. M., Taylor, E. W., Kuntz, I. D., Gibson, B. W., and Dollinger, G. (2000) High throughput protein fold identification by

using experimental constraints derived from intramolecular cross-links and mass spectrometry. *Proc.Natl.Acad.Sci.U.S.A* **97,** 5802-5806.

31. Zhang, Z. and Marshall, A. G. (1998) A universal algorithm for fast and automated charge state deconvolution of electrospray mass-to-charge ratio spectra. *Journal of the American Society for Mass Spectrometry* **9,** 225-233.

SUPPLEMENTARY MATERIAL

Supplementary material relating to the development of AnchorMS and the compilation of this manuscript, including source code and calibration data, is available online at the following URL:

cbio.ufs.ac.za/AnchorMS/Development

SUMMARY

Multi-subunit protein complexes are involved in many essential biochemical processes including signal transduction, protein synthesis, RNA synthesis, DNA replication and protein degradation. An accurate description of the relative structural arrangement of the constituent sub-units in such complexes is crucial for an understanding of the molecular mechanism of the complex as a whole. Many complexes, however, lie in the mega-Dalton range, and are not amenable to X-ray crystallographic or Nuclear Magnetic Resonance analysis. Techniques that are suited to structural studies of such large complexes, such as cryo-electron microscopy, do not provide the resolution required for a mechanistic insight.

Mass spectrometry (MS) has increasingly been applied to identify the residues that are involved in chemical cross-links in compound protein assemblies, and have provided valuable insight into the molecular arrangement, orientation and contact surfaces of sub-units within such large complexes. This approach is known as MS3D, and involves the MS analysis of cross-linked di-peptides following the enzymatic cleavage of a chemically cross-linked complex. A major challenge of this approach is the identification of the cross-linked di-peptides in a composite mixture of peptides, as well as the identification of the residues involved in the cross-link. These analyses require bioinformatics tools with capabilities beyond that of general, MS-based proteomic analysis software. Many MS3D software tools have appeared, often designed for very specific experimental methods. We review all major MS3D bioinformatics programs currently available, considering their applicability to different workflows, specific experimental requirements, and the computational approach taken by each.

We also developed AnchorMS, a new bioinformatics tool for the identification of both the sequences and cross-linked residues of di-peptides within a post-digest peptide mixture based on MS^1 and MS^2 data. AnchorMS is intended as a component in the workflow of an MS3D experiment where the protein sequences, cross-linking reagent and protease are known.

AnchorMS is freely available as a public web service at cbio.ufs.ac.za/AnchorMS via a simple, user-friendly web interface coded in PHP/XHTML. Experimental sample preparation information and MS data may be uploaded through the web form and analysed by AnchorMS. After analysis, the web interface displays the di-peptides detected, as well as the calculated maximum inter-residue distance between cross-linked residues. This distance information can be used in the optimization of sub-unit positioning within structural models using third party software.

The computational core of AnchorMS was developed as an open-source Python project. We describe in detail the overall structure and workflow of the code as well as the functionality implemented in each section of the code.

AnchorMS creates a digital library of possible di-peptides and generates expected precursor and fragment mass spectra for each. In order to identify di-peptides, the observed mass spectra are matched against the library of expected mass spectra. Features that are unique to AnchorMS are highlighted, including those for the analysis of di-peptides where the sequences are identical, but the cross-linked residues differ. AnchorMS considers their possible co-fragmentation and employs a specialised second score for distinguishing between such precursors.

A unique mathematical model for estimating the level of false positive matching was derived based on an *in silico* simulation of false positive spectrum matching using randomly generated di-peptide sequences. Subsets of the simulation data were modelled using disparate functions, which were subsequently combined to yield a composite model that described expected false matching under various conditions. The refined calibration of this model against simulation data was performed using the R programming language. AnchorMS also implemented this model as a dynamic false positive threshold, where score values greater than the threshold were considered likely to be true spectrum matches.

OPSOMMING

Proteïen komplekse wat uit verskeie subeenhede bestaan, is by baie essensiële biochemiese prosesse betrokke, insluitend by seintransduksie, proteïensintese, RNA-sintese, DNA-replikasie en by die afbreking van proteïene. 'n Akkurate beskrywing van die relatiewe strukturele rangskikking van die subeenhede in sulke komplekse is van kardinale belang vir 'n begrip van die molekulêre meganisme van die kompleks as 'n geheel. Komplekse is egter dikwels groter as een megaDalton, en dus nie geskik vir X-straal kristallografiese of kern magnetiese resonansie analises nie. Tegnieke wat vir strukturele studies van sulke groot komplekse gebruik word, soos krio-elektronmikroskopie, gee nie die nodige resolusie vir 'n meganistiese insig nie.

Massaspektrometrie (MS) word toenemend toegepas om die aminosure te identifiseer wat in chemiese kruisbindings in die saamgestelde proteïen kompleks betrokke is, en verskaf waardevolle inligting oor die molekulêre rangskikking, oriëntasie en kontak oppervlaktes van subeenhede binne sulke groot komplekse. Hierdie benadering staan bekend as MS3D, en behels die MS analise van kruisgekoppelde peptied dimere na die ensiematiese klowing van 'n chemies kruisgekoppelde kompleks. 'n Groot uitdaging van hierdie benadering is die identifisering van die kruisgekoppelde peptied dimere in 'n saamgestelde peptied mengsel, sowel as die identifisering van die aminosuur residue wat in die kruisbinding betrokke is. Hierdie ontleding vereis bioinformatika sagteware met vermoëns buite dié van algemene, MS-gebaseerde proteoom analise sagteware. Baie MS3D sagteware het al verskyn, en is dikwels ontwerp vir baie spesifieke eksperimentele metodes. Ons gee 'n oorsig van die belangrikste MS3D bioinformatika programme tans beskikbaar, met inagneming van hul toepaslikheid in verskillende werk metodes, spesifieke eksperimentele vereistes, en die rekenaarmatige benadering wat elkeen neem.

Ons het AnchorMS ontwikkel, 'n nuwe bioinformatika program vir die identifisering van beide die volgordes en die spesifieke kruisgekoppelde aminosure van die peptied dimere binne 'n verteerde peptied mengsel, gebaseer op MS^1 en MS^2 data. AnchorMS is bedoel as 'n komponent in die werksvloei van 'n MS3D eksperiment, waar die proteïen volgordes, kruisbindende reagens en protease bekend is.

AnchorMS is vrylik toeganklik as 'n openbare webdiens by cbio.ufs.ac.za/AnchorMS via

'n eenvoudige, gebruikervriendelike webkoppelvlak, gekodeer in PHP/XHTML. Eksperimentele monster voorbereiding inligting en MS data word deur die webvorm opgelaai, en deur AnchorMS geanaliseer. Na die analise, vertoon die webkoppelvlak waargenome peptied dimere asook die berekende maksimum afstand tussen kruisgekoppelde aminosure. Hierdie inligting kan in die optimisering van subeenheid posisies in strukturele modelle gebruik word, met behulp van derdeparty sagteware.

Die rekenkern van AnchorMS is as 'n oopbron Python projek ontwikkel. Ons beskryf in detail die algemene struktuur en werksvloei van die program, sowel as die funksionaliteit wat in elke afdeling van die kode geïmplementeer is.

Soos met ander MS3D programme, stel AnchorMS 'n digitale biblioteek saam uit moontlike analiet peptied dimere en genereer vir elkeen verwagte MS spektra. Om peptied dimere te identifiseer, word die waargenome massa spektra vergelyk teen die biblioteek van verwagte massa spektra. Dié funksionaliteit, wat uniek tot AnchorMS is, word uitgelig, insluitend dié vir die analise van peptied dimere waar die volgordes identies, maar die kruisgekoppelde aminosure verskil. AnchorMS oorweeg ook hulle moontlike medefragmentasie, en pas 'n gespesialiseerde, tweede telling toe om tussen sulke gevalle te onderskei.

‘n Unieke wiskundige model is afgelei vir die beraming van die vlak van valse, positiewe identifikasies, gebaseer op 'n rekenaar simulasie van spektra voorspel vir peptiede met lukraak gegenereerde volgordes. Gedeeltes van die simulasie data is gemodelleer deur die gebruik van funksies wat gekombineer is om 'n saamgestelde model te lewer, wat die verwagte vlak van valse, positiewe identifikasies onder verskillende omstandighede kan beskryf. Verfyning van hierdie model is met behulp van die R programmeringstaal uitgevoer. AnchorMS implementeer ook hierdie model as 'n dinamiese, valse positiewe drempel, waar telling waardes groter as hierdie drempel beskou word as korrekte identifikasies.

KEYWORDS

Bioinformatics

Cross-linking

Interactions

Mass spectrometry

Modelling

MS3D

Protein

Python

Software

Structure